CAS MONOGRAPH SERIES
NUMBER 16

FROM GLMs TO COMPREHENSIVE INSURANCE PRICING

Techniques and Challenges

Alan Chalk
David Deacon
Montserrat Guillen
Max Martinelli

Commissioned by the Exam 8 Working Group

CASUALTY ACTUARIAL SOCIETY

From GLMs to Comprehensive Insurance Pricing: Techniques and Challenges

Alan Chalk
David Deacon
Montserrat Guillen
Max Martinelli

Casualty Actuarial Society
4350 North Fairfax Drive, Suite 250
Arlington, VA 22203
www.casact.org
(703) 276-3100

From GLMs to Comprehensive Insurance Pricing: Techniques and Challenges
By Alan Chalk, David Deacon, Montserrat Guillen, Max Martinelli

ISBN (print edition) 978-1-7370028-8-8
ISBN (electronic edition) 978-1-7370028-9-5

Contents

Acknowledgments

As always, with a work of this size, there are many acknowledgments.

To the team at CAS, whose idea this monograph was: Thank you for kicking off the process and supporting it all the way through.

To the Exam 8 Syllabus Committee, Brian Ironside and Howard Mahler: Your reviews, feedback, and challenges on the various drafts were immensely useful in improving both the readability and content of the paper. Many thanks also for correcting all my grammatical errors. Though there is one thing you have not cured me of. Sentence fragments.

To my co-authors—Dave Deacon, Montse Guillen Estany, and Max Martinelli—and also to Tom Holmes and Suguru Fujita: It has been a pleasure and an immense privilege working with you. Your wisdom and knowledge are vast, and your insights and input were invaluable.

To Yijia Liu and Luís Sousa: Thank you for being amazing and patient editors.

Finally, to Joana Amorim at ACTEX: Your constant support and encouragement are the only reason we got this over the line. Thank you.

Alan Chalk
May 2025

About the Authors

Alan Chalk is an Actuarial Data Scientist at Sabre Insurance Limited, where he applies actuarial and machine-learning techniques to enhance pricing and fraud-detection capabilities. He has many years of experience across a broad range of actuarial practice and has previously held various roles in the industry, including Global Aerospace Actuary at AIG, Chief Actuary at Groupama and Allianz Marine and Aviation, and Group Pricing Manager at RSA.

David Deacon has more than 20 years of experience in the insurance industry. He is responsible for the pricing, product development, underwriting strategy, and strategic initiatives for Heritage Insurance Holdings. Prior to joining Heritage, Dave held various actuarial and product roles at Hanover Insurance and MetLife Auto & Home. Dave holds the professional certifications of Associate in the Casualty Actuarial Society (ACAS), Member of the American Academy of Actuaries (MAAA), Chartered Property Casualty Underwriter (CPCU), and Associate in Reinsurance (ARe). He served as an adjunct professor at Bryant University, teaching an undergraduate course on property and casualty reserving. Dave received his bachelor's degree in mathematics from Worcester Polytechnic Institute.

Montserrat Guillen is a Full Professor of Quantitative Methods at the University of Barcelona and leads the UBriskcenter research group. She is recognized as a top global author in telematics insurance and is the most-cited woman in actuarial science worldwide. She currently serves as Chief Editor of the *North American Actuarial Journal*. She has supervised over 20 PhD theses and has participated in international doctoral evaluations and committees in France, the UK, the US, Canada, and Belgium. She has received major research awards, including from the Casualty Actuarial Society and the International Insurance Award. She is an active member of key actuarial and statistical association and is a prominent speaker at global academic and industry events.

Max Martinelli is a Lead Actuarial Data Scientist at Akur8, where he works with insurers across the North American property and casualty market to apply machine learning to

pricing and rating. He has over a decade of experience in actuarial and data science roles, primarily in predictive modeling for personal and commercial lines, and a background in machine learning and computational mathematics. A dedicated collaborator with the Casualty Actuarial Society, he co-designed and led the CAS AI Fast Track bootcamp and co-hosts *Almost Nowhere*, the CAS Institute podcast exploring AI and data science in insurance.

1. Introduction

1.1. Objectives

The CAS monograph *Generalized Linear Models for Insurance Ratings* by Goldburd et al. (2025) is a comprehensive guide to creating insurance rating plans with generalized linear models (GLMs). It covers the technical foundations of GLMs and practical topics, such as the model-building process, data preparation, selection of model form, model refinement, and model validation.

This monograph assumes that the reader is familiar with Goldburd et al. (2025)[1] and uses case studies to focus on various practical challenges in building GLMs. Many of these challenges, along with suggested solutions, have been mentioned in the CAS GLM monograph. Here, we expand on and demonstrate these solutions using the case studies and discuss some additional practical challenges.

These practical challenges (which will be explained in more detail later on) include:

- Dealing with high cardinality categorical variables (HCCVs).
- Dealing with situations where part of the rating plan has to be calculated separately from the main GLM (for example, geographical analysis). In these situations, we may need to add back to our GLM the results of the separate analysis and then do further GLM-based work.
- Removing from a GLM the effect of variables we are not permitted to use or that we do not want to use. Here we will have to consider whether we allow ourselves to use variables that are correlated with the forbidden variable.
- Dealing with highly correlated variables.

The case studies will require us to revisit various basic ideas discussed in the CAS GLM monograph, including:

- How to measure model performance.
- Splitting the data into training and test sets.
- Dealing with nonlinear effects.

[1] ...and with the relevant chapters of *An Introduction to Statistical Learning* (James et al. (2021)).

Penalized regression is an approach to fitting GLMs that is impossible to ignore. Section 10 of Goldburd et al. (2025) discusses it and mentions that the main disadvantage is that it is "much more computationally complex than standard GLMs." On the other hand, the advantages of using it are significant. The reader will see throughout the text that it is easy within the penalized regression framework to ensure that models are not overfitted and therefore perform well on future data (it will also make our life much easier when we develop the case studies). Regarding the issue of computational complexity, while this can be an issue, significant computing power is easily available to us today, either with high-specification machines running locally or in the cloud. Open-source software exists that fits GLMs (and other models) on datasets which are too large to fit into memory. For those who prefer not to write, debug, and maintain code, proprietary software exists that makes it very easy to fit good penalized GLMs. Overall then, we really have no excuse not to use penalized regression. We devote one chapter to reviewing its basic ideas and some theory before applying it in subsequent chapters. Many of the practical concepts above apply equally well whether one uses GLMs with or without penalization. (The relevant R or Python code is also fairly similar.) Therefore, the determined reader who wishes to remain in the "without penalization" world will be easily able to do so.

One word of warning: we discuss methods in this paper that seem automatic and to reduce the level of human input in model training in general and in the setting of insurance rating plans in particular. The reality is that most of the time, these methods simply allow the practitioner to make more informed choices and avoid certain pitfalls. The potential negative consequences of mindlessly creating rating plans without human input and review remain as large now as they ever have been.

2. The Case Studies

2.1. FAA-NTSB Data for Aircraft Incident Frequency

In the US, the Federal Aviation Administration (FAA) maintains records of all registered aircraft, and the National Transportation Safety Board (NTSB) maintains a database of all incidents (accidents and near misses) involving aircraft. Our main case study will be based on these records. Our dataset consists of all aircraft from the FAA records. Each aircraft is represented by one or more lines of data, one for each calendar year, depending on its certificate issue date and expiration date. For example, if the certificate issue date is July 1, 2010, and the expiration date is December 31, 2014, then the aircraft will be represented by five lines of data, one for each calendar year from 2010 to 2014. The exposure variable (named "ex" in our data) will be 0.5 (i.e., the six months from July 1, 2010 to December 31, 2020) for the first year and 1 (year) for the others. We added a target variable (number of claims, named "num_cl" in our data) set to 1 where there was a match to an accident on the NTSB data and set to zero otherwise. Our aim will be to build a frequency model that predicts the probability of an aircraft accident (given the information we have about the aircraft, its use, etc.).

Although our focus will be on building a frequency model, almost everything we discuss applies equally to severity models. We note also that sometimes in aviation, rates for total loss insurance are supplied separately. Provided that the policy is on a declared value basis,[2] and the rate is charged per dollar of declared value, a severity model would not be needed. For example, our final model predicts a frequency of about 0.6 accidents per 1,000 aircraft for the Cessna Skyhawk. If about a third of these accidents are total losses, then an estimate of the total loss rate would be 20 cents per $100.[3]

NTSB accidents since 2008 are used, and only aircraft with some active registration since that year are included. The FAA data includes the registration number (N-number) for each aircraft and has fields for certificate issue date and expiration date. We merged

[2] A declared (or agreed) value basis means the insurer and policyholder agree in advance on the amount payable in the event of a total loss, regardless of the actual replacement cost or market value at the time of the claim.

[3] If one was using the NTSB data for such purposes, one would need to remember that total loss incidents can happen without being reported to the NTSB.

the claims where the FAA N-number matched the NTSB N-number and the date of the NTSB incident was between the FAA certificate issue date and expiration date. On this basis, about 50% of claims were merged into the FAA data. Inspection of the unmatched incidents shows that many of them are for foreign aircraft not registered with the FAA. Were we to be using this data as part of a real rating exercise, we would look into this in further detail. However, for our purposes, the resulting dataset is sufficient as it allows us to illustrate and discuss various practical issues when fitting GLMs. For example:

- There are many different makes of aircraft, which leads to practical problems when trying to create a rating plan which has make of aircraft as a rating factor.
- The accident frequency varies with aircraft age[4] but not in a linear way—see Figure 2.1.

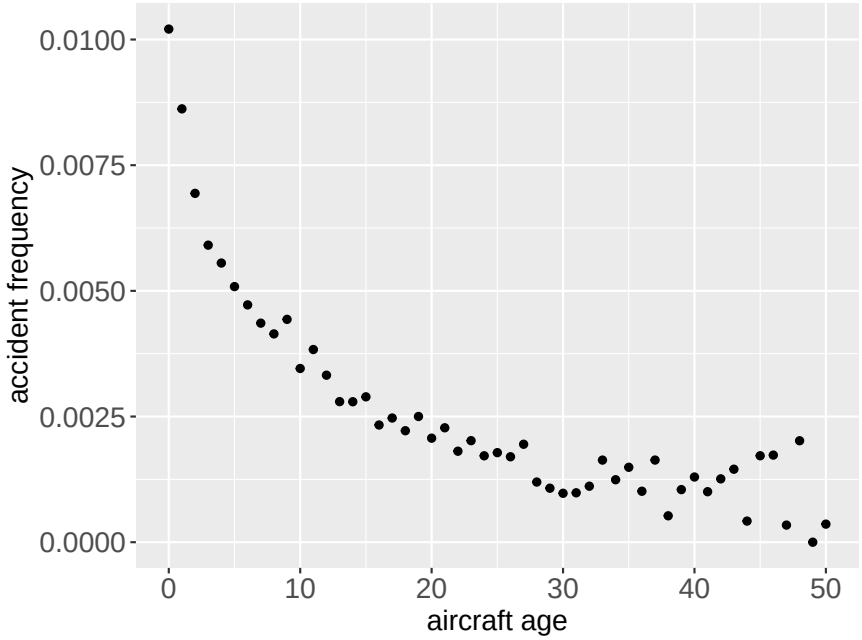

Figure 2.1. FAA-NTSB data. The frequency of accidents varies with the age of the aircraft, but not in a linear way.

An example of some of the data in the FAA-NTSB dataset is shown in Table 2.1. (Note that across all of our observations, N-number is not a unique identifier of an aircraft, because once an aircraft is deregistered, the same N-number can be used for a new aircraft.)

[4] Throughout this monograph, when we use the term "aircraft age," we actually mean the number of years since the FAA certificate was issued. This is an error as these two features are not the same. We discuss this matter further in Section 10.1.

Table 2.1. A sample of information in the FAA-NTSB data. There is a separate line of data for each aircraft for each calendar year. For example, the second aircraft in the table will have eight lines in the data, from age 0 to age 7. ex is the exposure time (the proportion of the calendar year during which the aircraft was operated). num_cl is the number of claims (none of these records was matched to an accident on the NTSB data).

n_number	city	aircraft make	aircraft age (years)	ex	num_cl
N106K	KONAWA	BEECH	25	1.00	0
N64882	GREELEY	CESSNA	7	1.00	0
N6407	ASHEBORO	HILLER	36	1.00	0
N6644V	COLLEYVILLE	BELLANCA	5	0.66	0
N9271W	SOUTH RIDING	PIPER	11	1.00	0

Finally, we note that for the purpose of quicker fitting of our models, we used only a 25% random sample of this dataset.

2.2. Road Accident Frequency Model

In the UK, the police maintain a record of every road traffic accident involving injury which they attend. The dataset is known as STATS19, and the UK government makes it available for download in order to encourage research into road safety. It is (much more) conveniently available through an R data package called stats19 (see Lovelace et al. 2019) available on CRAN.[5] We summarized this data by geographic region and added some fields from the UK census which provide sociodemographic features. Our aim will be to build a model which predicts the probability of road traffic accidents in a given region.[6]

The data preparation process led to this dataset including many highly correlated variables. Table 2.2 shows examples of some of the high negative correlations. We will see that this presents practical difficulties when we try to fit a model.

[5] https://cran.r-project.org/web/packages/stats19/index.html

[6] Such a model may be useful in helping to target road safety programs and expenditure.

Table 2.2. Some of the high negative correlations present in the road traffic accident data. The meanings of some key variables in this dataset will be explained as we progress through this case study (see the footnote for an example).

variable 1	variable 2	cor
c_ts011_hhd_0_pct	c_ts011_hhd_3_pct	−0.86
c_ts002_mcp_never_pct	c_ts002_mcp_mrcp_pct	−0.92
c_ts002_mcp_never_pct	c_ts002_mcp_m_pct	−0.92
c_ts002_mcp_never_pct	c_ts002_mcp_mos_pct	−0.92
c_ts011_hhd_0_pct	c_ts011_hhd_2_pct	−0.95
c_ts004_cob_uk_pct	c_ts004_cob_oth_pct	−0.96

2.3. Simulated Data—Auto Insurance

We have simulated data to imitate some aspects of frequency rating for auto insurance.

When simulating the data, we decide in advance what the true underlying frequencies will be. For example, we can decide that the starting point will be a frequency of 5% and that risk is increased by 4.65 times for a 17-year-old male and 2.55 times for a 17-year-old female. We call these true underlying frequencies and relativities the "ground truth." As part of our work on case studies using real data, we will be choosing performance metrics to measure model quality. It is not always obvious what a good or bad value is for a performance metric, and we will use these data to illustrate the values we might typically expect.

2.4. The Challenges

2.4.1. High cardinality categorical variables (HCCVs)

In the FAA-NTSB dataset, the rating factor "make of aircraft" (Cessna, Piper, Beech, etc.) is a categorical variable. It divides our data into groups or categories, and these groups do not have a natural order or ranking. The CAS GLM monograph explains that when using a GLM (not penalized), the prediction for any category is the average claims experience for that category. There are many different makes of aircraft, and given the overall moderate size of the dataset there may be some makes which have no accidents in our data. If we include aircraft make as one of our rating factors, then our model will predict a frequency of 0% for such makes. This is not a reasonable prediction. Also,

[1] As an example, `c_ts002_mcp_never_pct` and `c_ts002_mcp_mrcp_pct` both address marital and civil partnership status. The former is the percent of the population in a region who were never married and never registered a civil partnership, and the latter is the percent of the population who are married or in a registered civil partnership. We clearly expect these to be negatively correlated.

any category which does not have sufficient data for the observed claims frequency to be sufficiently credible will potentially have a predicted claims frequency which is much too high or much too low. In addition to unreliable predictions, we will see that in certain approaches to modeling, including a categorical variable with many features will lead to significant computational complexity.

Categorical variables with a large number of levels, to the extent that they lead to the type of problem above (given the size dataset being used), are known as high cardinality categorical variables (HCCV), and we will discuss how to deal with the challenges that they present.

2.4.2. Combining other models with a GLM

In our FAA-NTSB data, we want to end up with a rate for each make of aircraft. As discussed above, the make of the aircraft is an HCCV. Some techniques used for dealing with HCCVs are quite different from GLMs and are most easily done in a separate exercise. One approach therefore is to do some initial modeling with the GLM, but then to leave the GLM behind and do some further modeling, and finally to combine our non-GLM modeling back into the GLM. Even where HCCV modeling of make and model of aircraft can be done within the framework of one single GLM, we may wish as a second stage to adjust the rates based on underwriter or other input. In such cases, we will need to integrate our changes back into the GLM.

Geospatial smoothing (something we might want to do whenever we think that the level of risk depends on location) presents a similar issue. A GLM cannot directly do geospatial smoothing. We may therefore wish to start off with some GLM modeling, once again leave the GLM behind and fit a different type of model, and finally combine the results back with the GLM.

Another example is credit modeling. In a rating plan, the rates might depend on features related to an individual's credit rating. We have a choice; we can include all the credit-related features in the main model, or we can create a separate "credit score" model and use the score as a feature in the main model. Separate credit score modeling has various advantages. For transparency, we might want to know the total impact that credit-related features have on the rate, and this is very easy to know if a credit score is used. In addition, score variables, such as credit, group records in ways that might not otherwise be obvious from the underlying features. This allows for parsimony; the main model can use fewer features. In addition, it allows us to more easily discover and estimate interactions, since there might be strong interactions with the credit score that are not apparent with the individual components of the credit score.

2.4.3. "Controlling" for certain features

In our case studies and other examples, we will sometimes need to isolate the effect of one or more features on a target variable. Once we know the effect, we might need to ignore it (if it is a disallowed rating feature) or adjust it. We will discuss various approaches and their impact on the final rates.

2.4.4. Highly correlated variables

Goldburd et al. (2025) Section 2.9 explains that when the correlation between any two features is large, the coefficients fitted by a GLM can swing wildly in response to small changes in the data, and the standard errors associated with those coefficients will become large. They propose two approaches to dealing with this; removing all but one of any group of highly correlated predictors and preprocessing with principal component analysis (PCA) or similar techniques. We will encounter highly correlated features in one of our case studies and will use the opportunity to discuss the challenges posed by highly correlated features more broadly and to look at the possibility of using penalized regression to resolve them.

2.5. Next Steps

We have now seen the aims of our case studies—essentially in each case to build the best frequency model that we can given any constraints. We also understand some of the challenges that we will face. However, before we start modeling we need to think about performance measurement—and we look at that in the next chapter.

3. Performance Measurement

Section 6 of the CAS GLM monograph discusses measures of model fit and comparing candidate models. Likelihood and deviance are mentioned, as well as penalized measures of fit (Akaike information criterion [AIC] and Bayesian information criterion [BIC]).

The authors note that comparing deviance across different datasets is not feasible. This is because deviance, calculated and summed for each observation, results in larger values for datasets with more observations. (Deviance can also not be compared across models with different error distributions, as it depends on the assumed distributional form.)

They explain that penalized measures of fit (AIC, BIC) are important. This is because without penalization, adding more features to a model always reduces its deviance (when measured on data on which the model was trained). For instance, by including a unique identifier as a feature in a generalized linear model (GLM), you could achieve a zero deviance, but this is not a true measure of the model's effectiveness.

The approach we have chosen to use for this note is to use a measure that does not depend on the number of observations (pseudo-R^2) and to calculate it over data not used in fitting the model (validation and test data).

3.1. Pseudo-R^2

In linear regression, the R^2 measure quantifies the proportion of variance in the dependent variable that is explained by the independent variables. The formula for R^2 is

$$R^2 = 1 - \frac{\text{Sum of Squares of Residuals}}{\text{Total Sum of Squares}}.$$

There are various problems with using R^2 to measure the performance of GLMs (see Cameron and Trivedi (2013)). Instead, measures called pseudo-R^2 are used. We now introduce the version we will use by way of an example.

We use y_i to refer to the variable we are trying to predict and $\hat{y}_i$ refers to the model prediction. The subscript i refers to each individual record. Example data is given in Table 3.1.

Table 3.1. Example data. Four observations: two with claims and two without. The model estimated probabilities are given in the third column.

unique_id	y_i	$\hat{y}_i$
1	0	0.2
2	0	0.4
3	1	0.6
4	1	0.8

Under the Poisson distribution, the probability of y_i if the Poisson mean is equal to $\hat{y}_i$ (λ or μ are often used for this parameter) is given by:

$$pr[y_i|\hat{y}_i] = \frac{\hat{y}_i^{y_i} e^{-\hat{y}_i}}{y_i!}.$$

Given that y_i are just our data and will not change from model to model, the denominator is constant and can be ignored in further calculations. (If we kept it in, it would cancel out in both the numerator and denominator of the expression we eventually derive.)

We will move forward by taking logs and working with the log-likelihood. (The motivation for taking logs in the context of our discussion is not obvious. However, more generally in the context of GLMs, it leads to mathematical calculations that are easier to work with and estimators that have useful statistical properties.) In the text below, all logs are to the base e.

Excluding the ignored constant, each individual contribution to the log-likelihood of our data given our model (l_{model}) is therefore:

$$\begin{aligned}\log_e pr[y_i|\hat{y}_i] &= \log_e(\hat{y}_i^{y_i} e^{-\hat{y}_i}) \\ &= y_i \log_e(\hat{y}_i) - \hat{y}_i.\end{aligned}$$

The l_{model} of the data in Table 3.1, given the predictions shown, is:

$$\begin{aligned}&0 \times \log_e 0.2 - 0.2 \\ +&0 \times \log_e 0.4 - 0.4 \\ +&1 \times \log_e 0.6 - 0.6 \\ +&1 \times \log_e 0.8 - 0.8 \\ =& -2.734\end{aligned}$$

The simplest model we could fit would have no features at all beyond an intercept and would just fit the mean. This is called the null model. The mean of y_i in the above data is 0.5, and the null model log-likelihood (l_{null}) is therefore:

$$\begin{aligned} &0 \times \log_e 0.5 - 0.5 \\ +&0 \times \log_e 0.5 - 0.5 \\ +&1 \times \log_e 0.5 - 0.5 \\ +&1 \times \log_e 0.5 - 0.5 \\ =& -3.386 \end{aligned}$$

The improvement in likelihood from the null model to the fitted model is therefore:

$$l_{model} - l_{null} = -2.734 - (-3.386) = 0.652$$

We can suggest that the "perfect" model would predict y_i for each observation. This is called the saturated model, and its likelihood ($l_{saturated}$) is

$$\begin{aligned} &0 \times \log_e 0 - 0 \\ +&0 \times \log_e 0 - 0 \\ +&1 \times \log_e 1 - 1 \\ +&1 \times \log_e 1 - 1 \\ =& -2 \end{aligned}$$

The best possible improvement in likelihood is from the null model to the saturated model:

$$l_{saturated} - l_{null} = -2 - (-3.386) = 1.386.$$

We can therefore say, that as a percentage of the best improvement in likelihood possible, the model explains:

$$\frac{l_{model} - l_{null}}{l_{saturated} - l_{null}} = \frac{0.652}{1.386} = 47\%.$$

This measure

$$\text{pseudo-R}^2 = \frac{l_{model} - l_{null}}{l_{saturated} - l_{null}}$$

is what we will use.

Remembering (as mentioned in the CAS GLM monograph Section 6.1.2) that deviance of a model is $D_{model} = 2 \times (l_{saturated} - l_{model})$, it is easy to show that:

$$\text{pseudo-R}^2 = 1 - \frac{D_{model}}{D_{null}}.$$

This now looks quite similar to the R^2 statistic that we started off with, where D_{null} represents the total variance and D_{model} represents the residual variance. We can say that in the same way that R^2 measures the percent of variance explained, pseudo-R^2 measures the percent of deviance explained.

One advantage of using pseudo-R^2 is that it leads to a number that modelers can remember. They can remember that typically a certain model can explain, say, 12% of the deviance. It tends to be quite clear whether or not a new model is performing well or not.

In Section 3.7.1 we will use our simulation study to help us understand what typical values we might expect pseudo-R^2 to take in an insurance context.

In Section 3.7.2 we briefly discuss why we might need to give each line in our data a different weight and how this affects the pseudo-R^2 calculation.

3.2. Train—Validate—Test

The risk of overfitting GLMs and the idea of splitting the data into training and test datasets is discussed in Goldburd et al. (2025), Section 4.3. Likewise, they discuss validation and cross-validation. We revisit this discussion and add some extra details.

The CAS GLM monograph required penalization of the performance metrics which were based on deviance, because deviance of a GLM measured on the training data will only ever decrease as more features are added. This will encourage models that are far too complex and do not generalize well to data that the model was not trained on.

This issue can be avoided by never saying that a model is good because our pseudo-R^2 measure looks good (i.e., high) on the data on which the model was trained. Instead we hold out some of the data (typically 30%[7]) as a test dataset. (This is also known as a test partition.) We say that a model is good if the pseudo-R^2 is good on the test dataset.

No result (or graph) from the test dataset can ever influence the model structure or features used or parameter estimates. If that happens, then it ceases to be a test dataset. The authors of this monograph have seen practitioners who, when faced with a difficult decision during model fitting, have a quick look at certain graphs or other metrics on the test dataset. In such cases, training decisions have been made by use of the test dataset and it is—by definition—no longer a test dataset, and good performance on it does not

[7] Others might use similar but somewhat different values.

mean that the model is good. There is at least one proprietary software that hides the test dataset from the practitioner while the model is being fitted. In a team where data are prepared by some colleagues and models are trained by others, it would be reasonable for the data preparation team to withhold the test dataset until the training is complete.[8]

Besides needing to know the performance of our final fitted models, we might also need to know the performance during the model fitting process. This is because, as we will see in this monograph, when fitting GLMs many decisions need to be taken when fitting GLMs, such as which features to include and how to allow for nonlinear effects. In addition, we will see that when using penalized regression there is always one critical parameter whose value has to be chosen based on performance while training the model. If possible, we want to take data-driven decisions, i.e., decisions which improve the performance of the model based on test data. However, we cannot do that. As soon as we start taking model training decisions on the test data, it is no longer a real test.

Our approach to all decisions taken during training is to use k-fold cross-validation. We explain this by example:

Imagine we have a dataset with 100 observations only. We set aside observations 71–100 for the test data. We divide the remaining data into seven parts. Each part is called a "fold." The seven parts we choose are observations 1–10, 11–20, etc. We then train seven models as follows:

- Model 1
  - In this model, observations 1–10 (fold 1) will be used for validation.
  - Train a model on all observations excluding 1–10, i.e., train on observations 11–70.
  - Predict the model on observations 1–10.
  - Calculate the pseudo-R^2 on observations 1–10.
- Model 2
  - In this model, observations 11–20 (fold 2) will be used for validation.
  - Train a model on all observations excluding 11–20, i.e., train on observations 1–10 and 21–70.
  - Predict the model on observation 11–20.
  - Calculate the pseudo-R^2 on observations 11–20.

and so on. The results might look something like Table 3.2.

[8] This is also done for online predictive modeling competitions.

Table 3.2. Seven models are trained. Validation performance is measured for each model using the validation data for that model. The average cross-validation performance is 0.10.

model	observations used for		pseudo-R^2
	validation	training	
1	1–10	11–70	0.10
2	11–20	1–10 and 21–70	0.11
3	21–30	1–20 and 31–70	0.09
4	31–40	1–30 and 41–70	0.10
5	41–50	1–40 and 51–70	0.12
6	51–60	1–50 and 61–70	0.08
7	61–70	1–60	0.10
average			0.10

Our performance metric will therefore be the average cross-validation pseudo-R^2 over k validation folds during training.

If we are faced with multiple alternatives to evaluate during training, we will calculate the average cross-validation performance for each one. We will then usually[9] use the option with the highest performance.

In the context of using cross-validation in place of a test (or holdout) dataset, Goldburd et al. (2025) Section 4.3.4, raise a problem with cross-validation. They explain that every training decision must be taken within the training phase of each of the cross-validation models. If features are "hand selected" with care and judgment, this process would need to be repeated, in our case, seven times—once for each fold. Since highly manual processes cannot be repeated so many times, they must be made before the data are split into folds—which then invalidates the cross-validation performance measurement.

The approach in our case studies is, in fact, to use automatic feature selection and engineering techniques and to do so within each of our seven models. This means that each of the models fitted in our cross-validation process may contain different features. Therefore, once the model training is complete, a final model must be fitted on all the data from all training folds using the decisions arrived at during cross-validation. The pseudo-R^2 can be calculated on the test data. This data was not used in any part of the training, and so performance on the test data should therefore be indicative of future performance of the model on unseen data. The use of automatic feature selection and engineering techniques does not mean that the use of care and judgment is now redundant. However, it does mean that the analysis can sometimes proceed to a later stage before manual adjust-

[9] There is room for some actuarial judgment.

ments are made, so that cross-validation performance measurement can be carried out at a later stage and help inform more of our modeling decisions.

3.3. Generalization Error

In the above discussion, we talked somewhat informally about "future performance of the model on unseen data." The formal term for the error made on future data by a model trained on our training data is "generalization error".

We proposed using cross-validation to help make modeling decisions and to estimate generalization error. This is a common and accepted technique across many areas of machine learning in situations where there is not sufficient data to separate out datasets for training, validation, and testing.

We will see later—and due to its importance we note now—that the performance we see after implementing our models may be very different from our estimated generalization error. The reason for this is as follows: Our cross-validation partitions are random samples of our data, and so each partition will have a similar distribution of the features. For example, if in one fold most policyholders are above 30, this is likely to be true across all folds. Overall then, our cross-validation estimate of generalization error is for a portfolio consisting mostly of policyholders over 30. If our modeling process leads to a new rating plan which significantly increases prices for policyholders above 30 years old and decreases prices for younger policyholders, then the distribution of the future unseen data will be very different from that in the validation folds, and actual performance may be different from expected. This will particularly be the case if there is a weakness in the new rating plan which savvy policyholders notice and take advantage of. A vital part of premium rating is therefore to monitor (in depth) the mix of business after implementation of the new rates. If the percentage of business written in one segment changes unexpectedly, then performance of that part of the model should be reviewed.

3.4. Null Models

A null model is the most basic model possible—often an intercept-only model, where the predicted frequency for each observation is the mean frequency.

For our simulated auto insurance data, we calculate the average frequency on training data, and the null model predicts this value for every row in our validation or test data.

For our FAA-NTSB study, the most basic model possible is one which just predicts the mean frequency on the training data for every aircraft in the validation or test data.

Under some performance metrics, performance of a null model can look very high. For example, in the FAA-NTSB data, the average accident rate is 0.4%. This is true for both the training and test sets. If model predictions are either 0 (no accident) or 1 (accident), we might decide to use accuracy (the percent of predictions that are correct) as a

measure. A very naive model which predicts 0 (no accident) for every craft will have a test accuracy of 99.6%. This might sound good, but the model is clearly of no use. Hence, before starting to train serious models, it is a good idea to create a null model to ensure that we understand what zero performance looks like. It also helps us to set up a basic pipeline for our model—training one model for each of the k-folds, calculating predictions and performance.

In our case, we should expect both training and validation (pseudo-R^2) performance to be 0% for a null model. This is because

$$\text{pseudo-}R^2 = \frac{l_{model} - l_{null}}{l_{saturated} - l_{null}},$$

and for a null model the numerator is zero, since the (fitted) model is the null model.

We check this for a small sample (10,000 observations) of the simulated auto insurance data. As discussed above, we first put aside 30% of the data for testing and divide the remaining data into seven folds. (We did this by dividing the data into 10 folds and using the last three for testing.) The folds were labeled $1, 2, \ldots, 7$. We then "trained" seven models as follows:

- Model 1
  - In this model, we used fold 1 for validation.
  - Find the mean frequency on folds 2–7. It is 0.0887.
  - Predict 0.0887 as the frequency for all the observations in fold 1.
  - Calculate the pseudo-R^2 for fold 1.
- Model 2
  - In this model, we used fold 2 for validation.
  - Find the mean frequency on folds 1 and 3–7. It is 0.0926.
  - Predict 0.0926 as the frequency for all the observations in fold 2.
  - Calculate the pseudo-R^2 for fold 2.

And so on. Table 3.3 shows the results for all folds, and the results are indeed mostly close to zero.

Table 3.3. Performance under the null model. Pseudo-R^2 is close to zero.

fold	deviance_model	deviance_null	proportion_explained
1	2903	2902	−0.0001
2	2954	2954	−0.0000
3	3095	3095	−0.0001
4	3040	3040	−0.0001
5	2861	2859	−0.0004
6	3117	3116	−0.0004
7	2926	2925	−0.0001

There is a problem here. Why are the results not exactly zero? The null model is an intercept-only model, which therefore predicts the mean. The model which we are fitting also predicts the mean. These two models are identical, so why is the deviance not identical? The reason is that we train our model, i.e., teach it to predict the mean, on the training data. We calculate performance based on the validation data, which will have a slightly different mean. This issue is discussed further in Section 3.7.4.

Although we will generally avoid looking at test performance until the end of the study, we also fit the null model on all the folds and find performance on the test data.

Table 3.4. Performance on the test data under the null model. As expected, pseudo-R^2 is zero.

fold	deviance_model	deviance_null	pseudo-R^2
99	8952	8952	−0.0000

We can see that performance of the model on the test data is also very close to 0%. In addition, we can see that the deviances are about three times higher than those for each fold. This is because the test fold contains three times as much data as each fold (30% of the total simulated data compared to only 10% in each validation fold).

Although we do not show the results here, we checked on our other case studies and confirmed that performance as measured by pseudo-R^2 is zero for the null model.

3.5. Baseline Models

At this stage, before trying to deal with any of the practical problems this monograph will address, it would be useful to train basic GLMs on the various datasets. The purpose of this is to enable us to test later whether our suggested solutions to the practical problems actually improve performance. The main case study we will focus on is the FAA-NTSB

data,[10] and therefore we would like in particular to show the results of a basic GLM on this data.

Examples of the features potentially available to use in our baseline GLM are as follows:

- Information about the aircraft, e.g., aircraft make and model, engine(s) make and model, maximum speed, number of seats, etc.
- Information about the use of the aircraft.
- Information about the owner of the aircraft, e.g., type of registrant (individual, corporation, etc.) and their location (region, country, city), etc.

Unfortunately, we were not able to fit a GLM when naively using all the features. This is because categorical variables like aircraft make and model have many different categories. As we will discuss in Chapter 6, a naive use of these leads to various issues. When we tried, there were memory problems or very long run times. For reasons explained later, even had we managed to fit these models they would have been unlikely to perform well on test data. We therefore excluded from the baseline model all categorical features with more than 20 categories. Even then, GLMs failed to converge. The reason for this is that some of the features in the model are linear combinations of other features. An example is whether an aircraft is licensed for carriage of cargo. Whether or not an aircraft in our dataset is licensed to carry cargo can be determined from a (linear) combination of airworthiness code and other operations it is licensed for. This is called multicollinearity and will be discussed briefly in Chapter 7. We therefore found and dropped the features causing the problem.

Having excluded the troublesome features, we were left with 22, of which seven were numeric and 15 were categorical. For a given categorical feature, a GLM will fit one parameter for each category other than the base level. For example "type_registrant" has nine categories, and so the GLM will fit eight parameters. (The reason why our GLM will not fit a separate parameter for the base level of categorical features is again due to multicollinearity. Whether or not type_registrant is 1 can be determined from type_registrant_2 to type_registrant_9.) Overall our GLM will fit 82 parameters for the 15 categorical features, so the model will have 89 parameters in total (besides the intercept).

Seven GLMs were fit (excluding data from one of the seven folds in turn). For each GLM, performance was measured for the six folds of data it was trained on (training performance) and separately for the one fold of data which was withheld (validation performance). The results are shown in Table 3.5, where we can see an average validation

[10] The full dataset used is available as an R library from GitHub at FAANTSB. After installation of the library, the help on the dataset (?dt_faa_ntsb_clean) provides a list of all the features and their meanings.

performance on the seven folds of pseudo-R^2 of 0.142 and the average training performance is 0.157.

Table 3.5. Performance under the baseline model. Average pseudo-R^2 on validation data is 0.140. Train performance is higher: 0.156. Some of the patterns fitted by the model are just noise in the training data and are not present in the validation data.

	pseudo-R^2	
fold	train	validation
1	0.153	0.136
2	0.151	0.124
3	0.156	0.143
4	0.165	0.145
5	0.148	0.135
6	0.149	0.138
7	0.169	0.159
mean	0.156	0.140

The GLM output contains, as usual, coefficients and *p*-values. Part of the output is shown in Table 3.6. Given that we are using a log-link[11] to create a prediction, we sum the coefficients to find the linear predictor, and we then take the exponent. For example, if the coefficients shown were the only ones in the model, the linear predictor for a corporation (type_registrant 3) with a five-year-old aircraft would be

$$-2.447 + 0.410 + 5 \times (-0.086) = -2.467,$$

and the predicted accident frequency would be

$$\exp(-2.467) = 0.085.$$

We can see some large *p*-values and one very large negative coefficient. If we were to continue using unpenalized GLMs, we would need to think about whether these are reasonable to have in our model and, if not, what to do about them. We also note that the impact of age on the linear predictor is linear in age—we will later consider whether or not this is reasonable.

[11] And the Poisson distribution, but that does not determine how we find the predictions given the fitted coefficients.

Table 3.6. Some of the coefficients and p-values from the baseline model. Some p-values are large, and the magnitude of one coefficient is very large. If we were to proceed with unpenalized regression, consideration would need to be given to how to deal with these issues.

	Estimate	Pr(>|t|)
intercept	−2.447	
type_registrant2	−0.375	0.2628
type_registrant3	0.410	0.0000
type_registrant4	0.086	0.6987
type_registrant5	0.378	0.0685
type_registrant7	0.425	0.0000
type_registrant8	−0.072	0.8563
type_registrant9	−12.756	0.9958
aircraft age	−0.086	0.0000

3.6. Benchmark Models

From our baseline model, we have an idea of the minimum performance that we should expect from our GLMs. At the start of a GLM rating analysis, it is also useful to know what is a good level of performance, so that if we get close to it, we might consider moving on to other projects, and if we exceed it, we should both be happy and double-check our work. We will discuss later (Section 8.2) that a challenge in using GLMs is that we need to be sure to feed them the right features—a topic we will discuss in detail. Indeed, we will create a machine learning pipeline, with penalized regression at its heart, which should do exactly that.

Tree-based machine learning models deal implicitly with nonlinear features and with interactions. Their disadvantage is that they are hard to interpret. This means that if we base rating plans on such models, it will be hard to explain in simple terms why the price for a policy is a given amount or why the price has changed following a change in the value of a feature. If all we want is to know what good performance looks like, then interpretability is not necessarily a major concern. A difficulty with training machine learning models is that there are often many parameters to fine-tune, and training them can become an exercise in itself. That being said, there are both open-source and proprietary software solutions that allow relatively easy fitting of machine learning models.

We chose an open-source implementation of gradient boosting whose default parameters are designed to give good performance "out-of-the-box" and which automatically deals with HCCVs. As usual, we trained one model for each fold to get cross-validation performance, and we also trained one overall model. Average cross-validation pseudo-R^2 is 0.165. In the authors' experience, well-trained gradient boosting models can some-

times outperform well-trained GLMs, so we would not necessarily expect our GLM to meet this level of performance, but we would hope to get close.

Beyond using alternative machine learning models to check on our GLM's overall performance, we can also use these models and their predictions to check on certain aspects of our model fitting. This will be considered further in Chapter 9.

3.7. Practically Speaking

3.7.1. Typical values for pseudo-R^2

Baseline model validation pseudo-R^2 for the aviation case study is 0.140. Is this good?

For the simulation data, we have the ground truth, so we know what perfect predictions are. If we measure performance based on these perfect predictions, we get a pseudo-R^2 of 0.051. Why does a perfect model have such a (seemingly) low performance? It is because pseudo-R^2 compares the part of the deviance that we explain to the deviance explained by predicting every observation at its actual outcome. Since the actual outcome has some randomness, we will never achieve a model that predicts perfectly on that basis.

The typical values of pseudo-R^2 that we can achieve on a particular dataset and target variable depend on the signal-to-noise ratio in the data. While we cannot quantify this, it is not a major concern as pseudo-R^2 is typically used for relative comparisons between models trained on the same target over a given dataset. In other words, for our aviation case study, asking whether on not 0.142 is good is not a useful question. Rather, we need to measure performance of the various models that we build, and we can then use our performance metric to choose the best performing model. That being said, some practitioners are fond of pseudo-R^2 exactly because when applied with care, it can also support inter-dataset comparisons.[12]

In a business context, what we should almost always know is the performance of the existing business model. Our aim is to create a rating plan which is more accurate than the existing one. Therefore, we can define the existing model performance as the benchmark we want to beat. If we are starting with a new model, then our metric allows us to rank models, and while we don't know up front what "good" is, by creating null and baseline model predictions as discussed above, we can ensure that our models are an improvement on a sensible starting position. As discussed above, one can find machine learning models that are relatively easy to train. These can be used to give an idea of the upper limit of good GLM performance.

[12] At the end of this monograph, we discuss some errors made in preparing and cleansing the data. Correcting these errors does indeed reduce performance levels towards those seen in the simulated data.

3.7.2. Weights

In datasets for frequency models, it is rare for each observation to represent the same amount of exposure. Even if all policies have a period of exactly one year, sometimes a policy may cancel midyear—in which case exposure will be less than one year—or there may be midterm adjustments for which a new line of data is created. In such situations, the pseudo-R^2 calculation does not change much; y_i and $\hat{y}_i$ in Section 3.1 are the annualized frequencies and the contribution of each prediction to the log-likelihood is multiplied by the exposure.

In slightly more detail, remembering that

$$\text{pseudo-R}^2 = 1 - \frac{D_{model}}{D_{null}},$$

the contributions of individual observations to the deviance are multiplied by exposure. If the individual contributions to deviance are $dmodel_i$ and $dnull_i$ and exposure for each observation is ex_i then

$$\text{pseudo-R}^2 = 1 - \frac{\sum_i ex_i \times dmodel_i}{\sum_i ex_i \times dnull_i}.$$

3.7.3. Creating folds

In Section 3.2 we said that we would divide our data into training and testing sets,and that when training models we would use k-fold cross-validation.

There are various ways to create the folds. In Python or R (or any other language), it is very easy to randomly sample your data without replacement. Or we can use packages which contain functions or methods to do this. Different packages will do things differently, and it is worth reading the documentation. For example, in R, the caret package will take care to generate balanced cross-validation groupings, i.e., that the frequency of claims is almost the same on the data you use for training, regardless of which fold is being left out.

Creating folds using random sampling can sometimes cause problems. If you forget to set the seed, then your analysis will not be reproducible. Similar problems arise if you are working with different software and the fold is created in one software and not the other. One simple solution is to use a unique_id for each record (such as the policy number) to create the fold. Then, even if the folds are for some reason lost, they can be recreated. If the dataset is changed slightly or updated, you can be sure that the folds will be recreated in a very consistent way.

The authors have found the derivation of the fold from the policy number to be practically very useful for the reasons described above. Where the policy whole number is not

numeric, typically one of the trailing digits is. Alternatively, a function that creates a number from any object (known as a hash function) can be used, and the modulus is then taken of the resulting number.

There is another reason why using the policy number to create folds may be important. We want cross-validation folds to be independent tests of our training models. If observations on a cross-validation fold are correlated with the training data, then we do not have an independent test. In insurance datasets, multiple observations often come from the same policy, either due to renewals of the same policy or multiple risks associated with the same policy (e.g., multiple insured vehicles in the same household). Such records will be correlated. If we allow these records to be in different folds and we see good performance, it might just be because the frequency of customer claims in the validation fold is similar to the same customer in the training fold. There is no guarantee that our model will perform well in trying to predict customers it has never seen before. This is a form of "leakage"—an idea which we discuss further in Section 6.3. Using the policy (or customer) number to create the fold variable will ensure that all observations associated with the same customer end up in the same fold.

Some examples of the above issue need more specific care. For example, fire claims in private auto insurance are relatively rare. However, it does happen occasionally that there is a large fire in a parking lot and many cars are total losses as a result of the fire. The resulting claims will likely be mostly in one region. If these policies are spread across all the folds, then a model which trains itself to give a very high coefficient to the region of the parking lot will have a deceptively good cross-validation performance and yet be close to useless in predicting future fire claims (as well as overpricing the region where the parking lot fire occurred). Another example of these high-severity, low-frequency events is homeowners insurance, where large catastrophes (e.g., hurricanes, wildfires, etc.) may impact a specific geography.

3.7.4. Rebasing predictions

In Table 3.3, our model was a null model which predicted the mean, yet the validation performance was not exactly zero. For example, for fold 1, applying the trained model to the validation fold gave a deviance of 385.88, yet the null deviance of that fold is 383.00, so the numerator in our validation performance metric, $l_{model} - l_{null}$, was not exactly zero.

The reason is that we train our (null) model, i.e., calculate the mean, on the training data, whereas l_{null} uses the mean of validation data itself. For any fold, the mean frequency is not the same on the training data as the validation data. Hence, performance is not exactly zero. We can see in general that the performance rounds to something small but negative. This is because l_{null} has the advantage of looking directly at the validation

data and therefore gets closer to its mean.

In the very simple case of the FAA-NTSB null model, for the first cross-validation model, the training data (folds 2–7) mean is 0.004579 and validation data (fold 1) mean is 0.004593. We could rebase the model predictions by multiplying each of them by the ratio of the means $\frac{0.004593}{0.004579} = 1.0031$. Then, the pseudo-$R^2$ would be focusing on the ability of the trained model to better segment different risks, rather than its ability to correctly estimate the overall level of risk. And then, of course, pseudo-R^2 would be exactly zero because the trained model, predicting always the mean, does not differentiate at all between risks (and neither does the null model).

In the case of our Poisson log-link model, for any set of predictions, if their mean is not the same as the target, we can improve the log-likelihood (and hence pseudo-R^2) by changing the intercept of the linear predictor (i.e., applying a multiplicative constant to the predictions) until the means are the same. To see this, we start with the log-likelihood, which was shown above to be

$$l_{model} = \sum_i y_i \log_e(\hat{y}_i) - \sum_i \hat{y}_i.$$

We will express the linear predictor for each observation as

$$\eta_i = \beta_0 + z_i,$$

where β_0 is the intercept (which is the same for all observations) and z_i is the rest of the linear predictor. Given the log-link, the log-likelihood can be rewritten as

$$l_{model} = \sum_i y_i(\beta_0 + z_i) - \sum_i \exp(\beta_0 + z_i).$$

To find the intercept which gives the maximum, we differentiate with respect to β_0:

$$\begin{aligned}
\frac{\partial l_{model}}{\partial \beta_0} &= \frac{\partial}{\partial \beta_0}\left(\sum_i y_i(\beta_0 + z_i) - \sum_i \exp(\beta_0 + z_i)\right) \\
&= \sum_i (y_i - \exp(\beta_0 + z_i)) \\
&= \sum_i (y_i - \hat{y}_i).
\end{aligned}$$

At the minimum, the above sum is zero, which means that the mean of the predictions is the same as the mean of the target. Hence, for any given set of predictions, if the mean of the predictions does not equal the mean of the target, the intercept can be

adjusted until it is, and the likelihood will improve.

In the case of our null model, rebasing makes sense. We know that performance should be zero and that differences in predicting the overall mean are driven by noise in the data and not any good or bad model performance. In the case of a more complex model, the ability to get closer to the mean of the validation data might be driven by a much more important issue. The mix of business by rating factor can change over time. If one model is better than another in understanding the differences in risk between different levels of rating factors, and the mix changes, then both at the risk level and at the overall level, performance will be better. This argument would suggest that when we are dealing with complicated models, we should not rebase. However, given that our validation folds are assigned at random, it would be very surprising if there were significant changes in the business mix between the folds.

In addition, a closer look at the means on training and validation data (not shown here) highlights another issue. The difference between these means can vary significantly and yet have only a small effect of pseudo-R^2. It therefore seems that pseudo-R^2 is not necessarily a good way to check whether models are on average predicting correctly.

One approach is to first rebase the predictions to be correct on average and only then to use pseudo-R^2 to measure how close the model is to the target on each individual observation. The separate question of whether the model predicts correctly on average is easily measured by seeing how much rebasing was needed. If we find that a model with a better pseudo-R^2 on average needs more rebasing, we should certainly investigate this further.

We note that the above discussion mirrors the separation between basic and classification ratemaking. The overall price level is determined according to basic ratemaking methods, with its analysis of loss development, loss trends, premium trends, etc. The classification rates are determined separately, and the resulting rating factors are then applied to the base rate. Since the task of classification ratemaking is not concerned with estimating the correct overall rate level, but rather how much higher or lower a specific risk class is compared to that base rate, our evaluation of the models which estimate the classification rate should not be impacted by differences between the average prediction and the average observed value.

3.8. Next Steps

We are almost ready to start addressing some of our practical issues. First, though, in the next chapter, we recap the ideas behind penalized regression and refit our base-case models using this approach.

4. Penalized Regression

4.1. Introduction

Generalized linear models (GLMs) are fitted by maximizing the likelihood on the training data. Adding an extra predictor will almost always increase the likelihood on the training data, even if it is not a useful predictor for future observations (policies sold in the future). If we fit GLMs looking only at the likelihood on the training data, we risk fitting complex models, full of useless predictors, which will be inaccurate on future data. Approaches have been developed which can be used after fitting GLMs to avoid this pitfall, for example, choosing the GLM with the lowest Akaike or Bayesian Information Criterion (AIC or BIC, respectively, see Goldburd et al. (2025) Section 6.2.2).

AIC is given by

$$AIC = -2 \times l_{model} + 2p,$$

where p is the number parameters in the model other than the intercept.[13] This is the equivalent of trying to make the log-likelihood on the training data (l_{model}) as big as possible, except that for every extra parameter, there is a penalty of 1 to be deducted from the log-likelihood. Section 1.8.5 of Wood (2017) shows that AIC is an approximation of how well the model estimates the true underlying patterns—and thus models chosen by minimizing AIC will predict well on future observations not seen in the training data.

Why might models with fewer parameters perform better on future data? Consider a very rare make of car—there are only 10 of them, and each is insured with a separate private-auto insurer for one year. One of them has an accident leading to a claim, and nine do not. If each company uses a GLM on its own training data, then one of the GLMs will predict a claim frequency of 100% and nine will predict a frequency of 0%. Across all the companies, the average of the 10 GLMs is exactly correct. Where the average prediction of a model is close to the correct value across all the training sets it could have seen, we say that it has low bias. However, for each company, the GLM has fitted itself to exactly the experience of that company. This is reflected in the variance of the 10 GLM predictions,

[13] Some sources include the intercept. This would not affect the comparison of models.

which is large ($\sigma^2 = 0.09, \sigma = 0.30$). Where models fit themselves exactly to random fluctuations in the training data, we say that they have high variance. Sadly in this case, the low bias does not help any individual company—the high variance means they either give insurance away free or they charge far too much.

If the companies choose their GLMs based on AIC, the result would probably be different as AIC would encourage the separate parameter for this car to be removed. The GLMs will predict for this make; the average frequency for all makes, say, is 5%. Now the GLMs suffer from bias, since the average prediction is wrong (5% instead of 10%), but the variance of the GLM predictions for this make is 0 (since they all predict the same). In this particular case, every company is better off inasmuch as all their predictions are closer to the truth. This is because while the simpler models have more bias (except when there are redundant or unnecessary covariates in the more complex models), they suffer less from variance. The possible improvement in performance by accepting more bias in return for less variance is known as the bias-variance tradeoff.

In this chapter, we will discuss penalized regression, which also involves the application of a penalty. However, there are some key differences in the two approaches.

- The penalty in AIC and BIC alters a diagnostic produced from a fitted model but does not directly alter the model's fit. In contrast, we will see that penalized regression applies a penalty in a way that directly influences the training of the model, resulting in different parameter estimates.
- AIC and BIC penalties are based on the number of parameters in the model, encouraging simpler models with fewer features. (We typically view models with fewer features as less complex.) However, the effects of penalties in penalized regression (which will be described in the following sections) can be more subtle. While they may reduce the number of features, they can also lead to models with coefficients closer to zero. Whether a practitioner would call these models less complicated will depend on the context. We will see in Chapter 7 that penalized regression can allow us to fit very complicated models—albeit with small coefficients—in situations where non-penalized GLMs could not be fit at all. On the other hand, in the method we use in Chapter 5 to deal with nonlinear effects, smaller coefficients lead to less complicated models.
- We will see that by combining penalized regression with cross-validation we can control the balance between simple and complex models and so manage the bias-variance tradeoff.

4.2. Types of Penalty

4.2.1. Ridge regression

In penalized regression, one common penalty is the sum of the squares of the coefficients in the model. We will use $l(\beta)$ to represent the log-likelihood of training data given the fitted model and $l_p(\beta)$ to represent the penalized log-likelihood. Typically, the intercept, β_0, is not penalized. The coefficients $(\beta_1, \beta_2, \ldots, \beta_p)$ and β_0 are now chosen to maximize

$$l_p(\beta) = l(\beta) - \lambda \sum_{j=1}^{p} \beta_j^2.$$

(Regarding λ, see the footnote.[14])

This is known as "Ridge regression." λ is a non-negative parameter which must be chosen. If it is zero, then a normal GLM will be fit. If it is exceedingly high, then adding even one parameter with a small coefficient will not increase l_p, and a null model (i.e., an intercept-only model) will be fit.

An alternative way to express Ridge regression is that we maximize the log-likelihood subject to the constraint that the sum of the squares of the coefficients must not exceed a certain amount, i.e.,

$$\text{maximize } l(\beta) \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq s.$$

If s is very large, then a normal GLM will be fit. If s is zero, then an intercept-only model will be fit (normally the intercept, β_0, is not included in the constraints). While this formulation is mathematically equivalent to the former, it is not generally used in practice. It does, however, emphasize how regularization controls how large the coefficients can be.

Somewhere along the spectrum between the (simple) null model and the (complicated) non-penalized GLM, might be a model which has smaller coefficients (and is potentially less complicated) and performs better on future observations due to the bias-variance tradeoff mentioned previously.

As a simple example of Ridge regression, we look at how accident frequency of aircraft decreases with age. We include only the aircraft age in the model. An unpenalized GLM (Poisson, log-link) fits the following model:

[14] Sometimes, to simplify the math, $\frac{\lambda}{2}$ is used. Software will then just double λ and the sequence of fitted models will be identical.

$$\begin{aligned}\text{log of accident frequency} =& \beta_0 + \beta_1 \times \text{ log of aircraft age} \\ & - 4.352 - 0.548 \times \text{ log of aircraft age}\end{aligned}$$

Taking the exponents of both sides gives:

$$\text{accident frequency} = 0.013 \times \text{aircraft age}^{-0.548}$$

We now train penalized GLMs with various values of λ. Figure 4.1 shows the relationship between the λ parameter (when we think of the penalized likelihood) and the s parameter (when we think of the likelihood constrained by the condition that the sum of the squares of the coefficients must be less than s). As we increase λ, we are in effect insisting that the sum of the squares of the coefficients must be smaller and smaller.

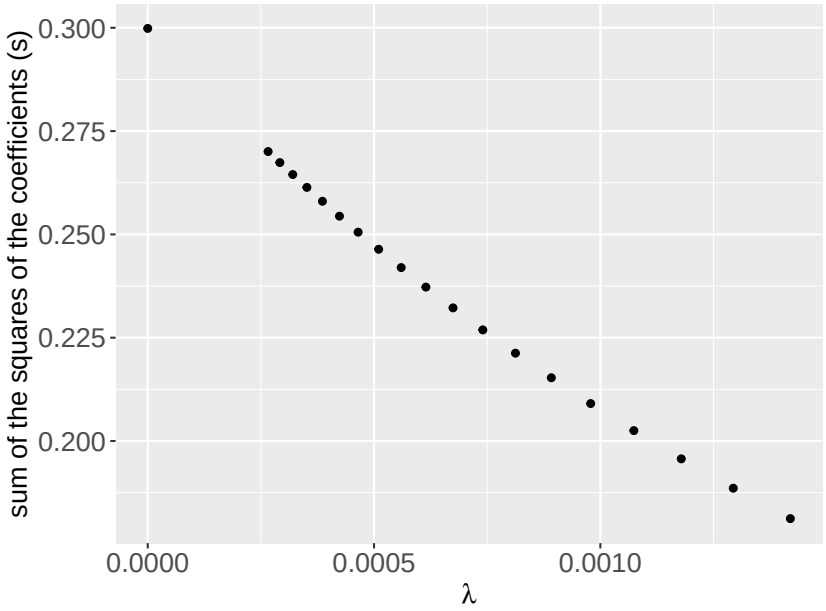

Figure 4.1. The relationship between the two different ways of looking at penalized regression. Asking for more penalization by increasing the value of λ is the same as insisting the the sum of the squares of the coefficients must be smaller.

Table 4.1 shows the results from penalized regression. If $\lambda = 0$, then as expected we get the same result as unpenalized regression. As λ gets larger, the coefficient shrinks towards zero. Are any of these alternative models for how accident frequency depends on

aircraft age more accurate than our original GLM? No. The third column is the mean cross-validation pseudo-R^2, and we can see that any significant amount of penalization reduces performance. In this case, there is no need for penalization to reduce model variance at the cost of increased bias because the model we are trying to fit is very simple.

Table 4.1. The log of aircraft age coefficient for various values of λ. When $\lambda = 0$, there is no penalization and the coefficient is exactly the same as the unpenalized GLM. The third column is the mean cross-validation pseudo-R^2. It can be seen that any amount of penalization reduces performance.

lambda	beta_1	mean CV pseudo-R^2
0.000	−0.548	0.0273
0.001	−0.487	0.0270
0.002	−0.417	0.0258
0.004	−0.305	0.0219
0.010	−0.182	0.0150
0.025	−0.090	0.0079
0.064	−0.039	0.0034
0.163	−0.016	0.0012
0.414	−0.006	0.0003
1.049	−0.003	−0.0001
3.192	−0.001	−0.0003

Ridge regression has other benefits besides performance. In the presence of highly correlated features, an unpenalized GLM will either fail to fit at all or the coefficients will be unstable. By "unstable" we mean that small changes in the training data can lead to large changes in coefficients. We will see in Chapter 7 that Ridge regression solves this problem.

4.2.2. LASSO

Another common penalty is the sum of the absolute values of the coefficients in the model (excluding the intercept). The coefficients $(\beta_1, \beta_2, \ldots, \beta_p)$ and the intercept β_0 are chosen to maximize

$$l_p(\beta) = l(\beta) - \lambda \sum_{j=1}^{p} |\beta_j|.$$

This penalty is known as the LASSO[15] and was proposed by Tibshirani (1996).

Table 4.2 shows the results of the LASSO on our simple FAA-NTSB model, where the accident frequency depends only on aircraft age. Once again, when $\lambda = 0$, as expected we get the same result as unpenalized regression. However, once λ gets large enough, the coefficient becomes exactly zero—it is removed from the model.

Table 4.2. The log of aircraft age coefficient for various values of λ. When $\lambda = 0$, there is no penalization and the coefficient is exactly the same as the unpenalized GLM. When λ gets large enough, the parameter is set to exactly zero (i.e., removed from the model).

lambda $\times 10^{-}3$	beta_1
0.0000	−0.548
0.0279	−0.542
0.1487	−0.517
0.9561	−0.352
2.4240	−0.050
3.1925	0

The shrinking of coefficients to exactly zero—effectively removing them from the model if λ is large enough—is a feature of the LASSO. It will happen if the benefit—when measured in a certain way—of including a feature in the model is less than a threshold controlled by the value of λ. Hence, as λ increases, more features will be removed from the model. For a given set of features and a value of λ the LASSO will select nonzero coefficients for the subset of features which are most beneficial to the predictive power of the model and will effectively exclude all the other features. Selecting some of the features and leaving out others is known as "feature selection." The fact that the LASSO does this itself during training and does not require the user to choose which features to include or exclude is known as "implicit" feature selection.

The LASSO is especially useful for datasets which have many features but only some of which are predictive. It will create a model with good performance while keeping it simple by excluding unhelpful features. The feature selection aspect of the LASSO will also work in the presence of highly correlated features but may not be stable, i.e., the features that it selects may differ with only small changes to the training data.

[15] The acronym stands for "least absolute shrinkage and selection operator." Tibshirani chose these letters to form the word "lasso," which has connotations of shrinkage and selection, and as a gentler alternative to "garotte," a term used by Leo Breiman for a related idea that inspired his work.

4.2.3. Elastic net

Zou and Hastie (2005) proposed a penalty which is a combination of Ridge regression and the LASSO, leading to

$$l_p(\beta) = l(\beta) - \lambda\left((1-\alpha)\frac{1}{2}\sum_{j=1}^{p}\beta_j^2 + \alpha\sum_{j=1}^{p}|\beta_j|\right).$$

This is known as "elastic net." α takes any value in the range [0, 1] and needs to be chosen. If $\alpha = 1$, this simplifies to the LASSO, and if $\alpha = 0$, this simplifies to Ridge regression.

We have discussed above that both Ridge regression and the LASSO can improve performance through use of the bias-variance tradeoff. In addition, Ridge regression helps fit stable models where our data contain highly correlated features, and the LASSO helps fit simple models in the presence of large numbers of unhelpful features. Where our datasets suffer from both issues, we can therefore consider the elastic net. From a practical perspective, given the risk that models with the LASSO penalty might be unstable, practitioners will often use elastic net with $\alpha = 0.99$ rather than the LASSO, even when the focus is purely on feature selection.

4.3. Choosing λ Using k-Fold Cross-Validation

In Ridge regression, or LASSO or elastic net, the optimal amount of penalization (i.e., λ) needs to be chosen. Given that the aim is to find the model which performs best on data which is not in the training data, we will use k-fold cross-validation, as described in Section 3.2.

We divide the data into k folds (partitions). In our case, we have seven folds labeled 1, 2, ..., 7. We remove the fold 1 and train the penalized regression model on folds 2–7. We don't know which value of λ to use, so we choose a sequence of (typically) 100 values of λ, ranging from a high value (which will tend to shrink all the coefficients to close to zero, leaving us with an intercept-only model) to a very low value of λ (low enough to give a result close to a non-penalized GLM). We now measure the performance (pseudo-R^2) for the models, one for each value of λ, on fold 1.

We apply this approach to the very simple case where the only feature is aircraft age. Figure 4.2 shows the pseudo-R^2 measured on fold 3 for the models trained on folds 1, 2, and 4–7. (The x-axis uses $\log(\lambda)$ rather than λ simply to space out the lambdas.) We can see that starting from the left of the x-axis, as λ increases, the performance on validation data first improves slightly and then decreases. This suggests that some penalization is useful in this case.

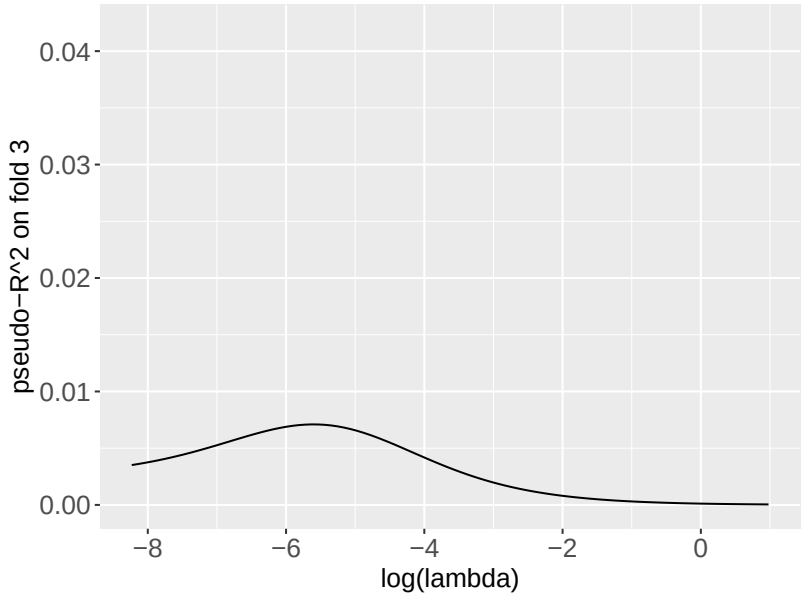

Figure 4.2. Performance of a model trained on folds 1, 2, and 4–7 as measured on fold 3. The x-axis uses $\log(\lambda)$ rather than λ simply to space out the lambdas which are created using an exponential scale. Starting from the left of the x-axis (low penalization), performance first improves slightly as penalization increases and then gets worse. The optimal amount of penalization is at $\log \lambda = -5.5$.

However, performance might vary for models trained and validated on different folds. We therefore repeat this exercise seven times, for example, training the model on folds 1, 3, 4, ..., 7 and then validating on fold 2, and then similarly for the other folds. Figure 4.3 shows the seven results—it can be seen that the results are different for each fold.

We therefore take the average performance across the folds for each value of λ. We can now see (Figure 4.4) that on average the validation performance gets worse as we increase penalization and, for this simple model, the optimal about penalization is zero.[16]

We have thus used k-fold cross-validation for choosing the optimal value for λ.

[16] It has been pointed out to the authors—many times—that a graph showing a case where there is no benefit to penalization is not the best way to introduce the reader to the benefits of penalization. However, even here there is a benefit of sorts—we know that the most complex model is the best performing model and that we are not overfitting the data.

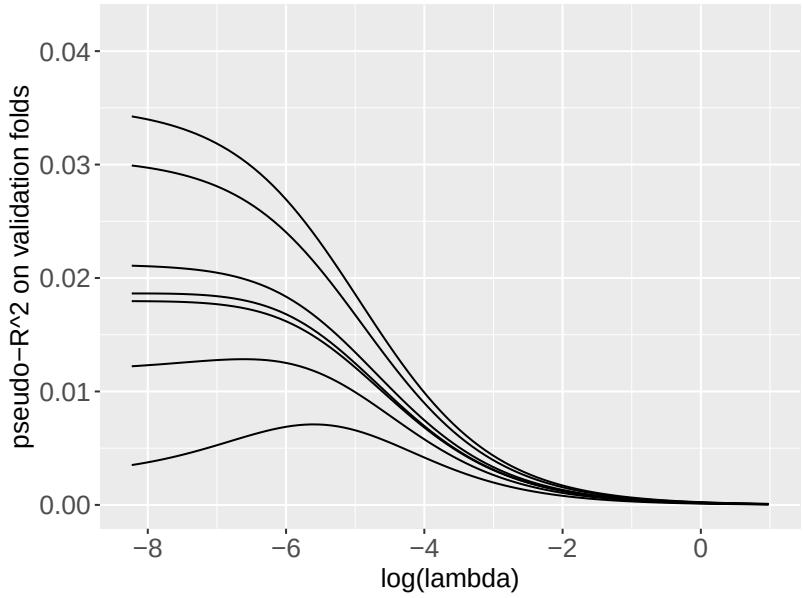

Figure 4.3. Validation performance for each of the seven folds. Though the validation curves mostly convey the same message—that for this simple model less penalization is better—the individual curves are different. We therefore take the average of these curves rather than rely on validation performance of one particular fold.

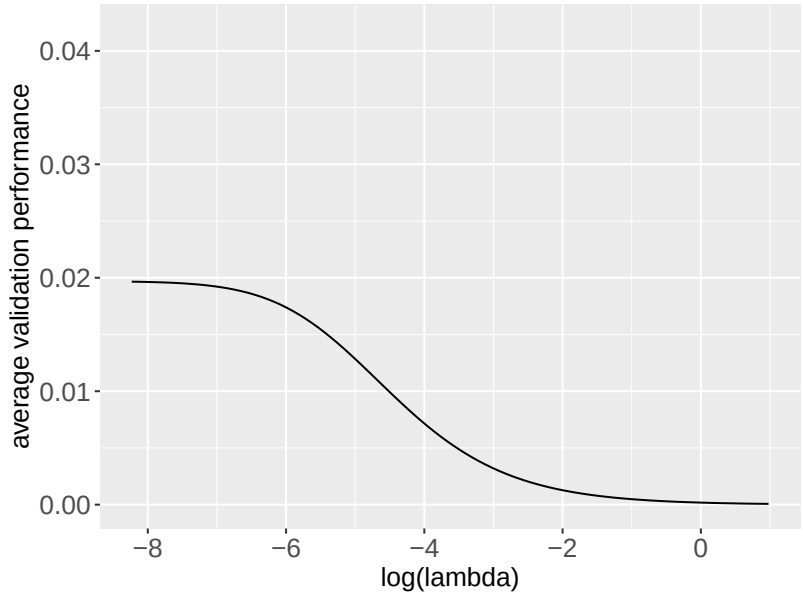

Figure 4.4. Average performance across the seven folds. The model with the best average validation performance is the model with no penalization (see footnote).

4.4. Case Study—Penalized Regression

In Section 3.5 we saw that the validation performance of a GLM on the FAA-NTSB data was 0.140. We now train penalized GLMs to see if we can improve performance by using the bias-variance tradeoff or if we can benefit from the feature selection aspect of the LASSO. (For the purpose of exposition, in this monograph we first fitted a non-penalized GLM and then moved to a penalized model. Penalized regression software typically returns models representing GLMs for a range of λs which include a model close to an unpenalized GLM. Therefore, it is unlikely that, in practice, a practitioner would ever need to run a separate unpenalized regression model.)

In order to investigate potential performance improvement and feature selection, we trained three models: Ridge regression, elastic net with $\alpha = 0.5$, and the LASSO. (As mentioned above, rather than the LASSO we actually use an elastic net with $\alpha = 0.99$ to encourage model stability.) In terms of performance, the three penalized models were similar, showing only a very small improvement over the unpenalized GLM. However, the LASSO does benefit from feature selection, and we discuss its results.

Figure 4.5 shows the validation performance for each fold. Some of the curves are slightly upward sloping to begin with, suggesting a small performance improvement for small penalization.

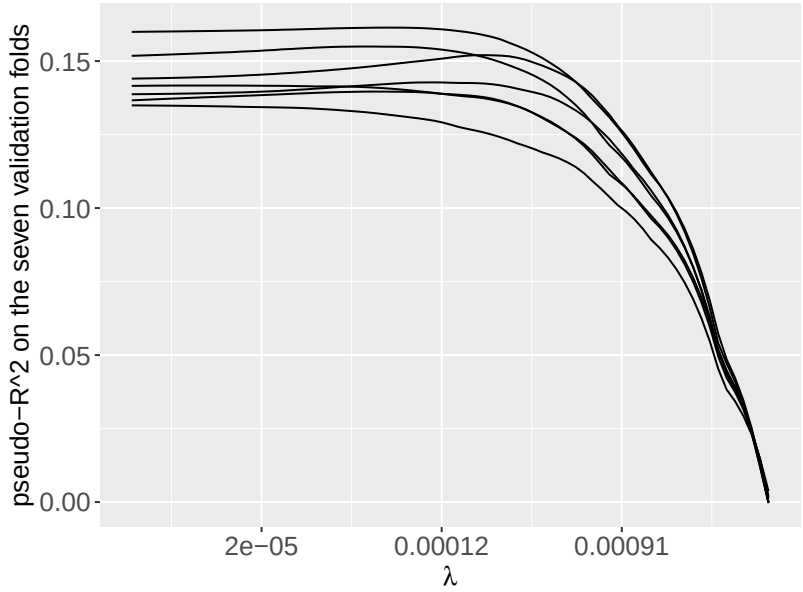

Figure 4.5. FAA-NTSB elastic net ($\alpha = 0.99$) on features excluding HCCVs. The performance on each fold is slightly different but at least some of them show a small benefit for some level of penalization.

Figure 4.6 shows the average validation performance across all the folds. The dashed vertical line is at the best performance—0.146—which is better than the baseline model (0.140). The numbers along the top are a count of the variables with nonzero coefficients. The best LASSO model benefits from having only 57 nonzero parameters compared to 89 in the non-penalized GLM previously fitted. (We actually gave the penalized regression model 105 features as we did not need to exclude collinear features. In addition, for every categorical variable we gave it a separate feature for each category, whereas in the non-penalized GLM we excluded the most frequently occurring category.)

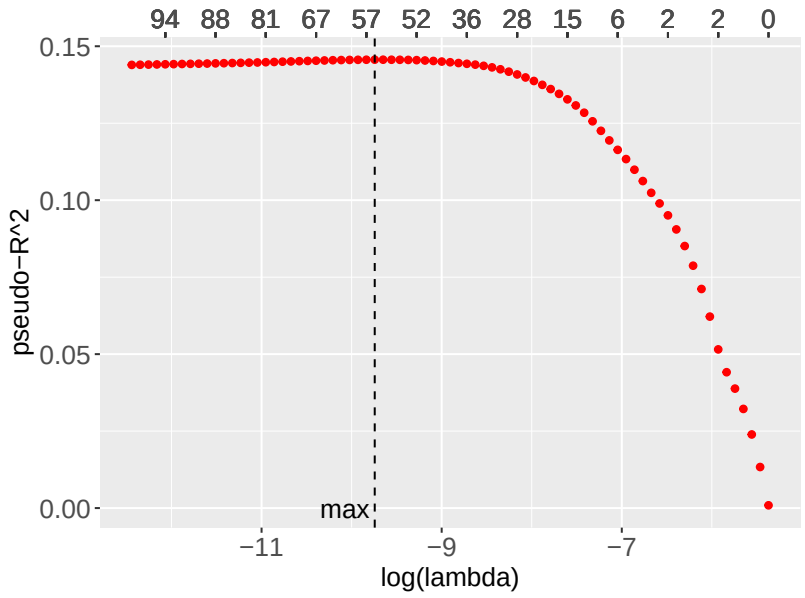

Figure 4.6. FAA-NTSB elastic net ($\alpha = 0.99$) on features excluding HCCVs. Average pesudo-R^2 performance across all the folds. The dashed vertical line is at the best performance.

Previously (Table 3.6) we saw the non-penalized regression coefficients. In Table 4.3 we compare them with the coefficients of our LASSO model. type_registrant9 and type_registrant4 which had high *p*-values in the non-penalized model have been penalized out completely. Other levels of type_registrant remain in the model, but we can see shrinkage (towards zero), which is generally greater when the *p*-values in the non-penalized model were high.

In the above case, the benefit of the LASSO was mainly its implicit feature selection (although there was a small performance benefit). In some cases, there can be clearer performance benefits. For example, Figure 4.7 shows the cross-validation results for the simulation case study. It can be seen that, starting from very low levels of penalization (on the left), performance improves as penalization increases, but then decreases.

Table 4.3. A comparison of the fitted coefficients of an unpenalized regression and the LASSO. type_registrant9 and type_registrant4 which had high p-values have been penalized out of the model. Other levels of this category remain in the model, but we can see shrinkage (towards zero).

	unpenalized coefficient	Pr(>|t|)	LASSO coefficient
type_registrant2	−0.375	0.2628	−0.194
type_registrant3	0.410	0.0000	0.392
type_registrant4	0.086	0.6987	.
type_registrant5	0.378	0.0685	0.288
type_registrant7	0.425	0.0000	0.390
type_registrant8	−0.072	0.8563	0.048
type_registrant9	−12.756	0.9958	.

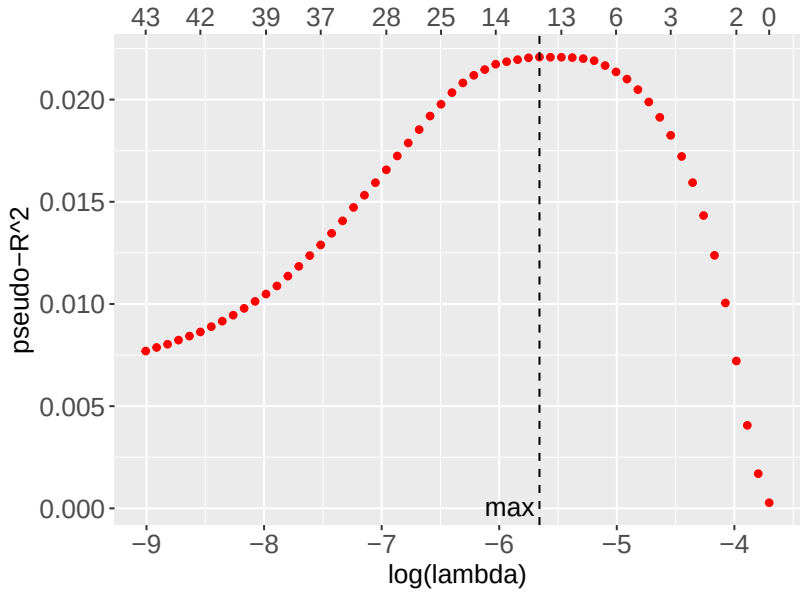

Figure 4.7. Simulation case study elastic net ($\alpha = 0.99$). The LASSO provides not only implicit feature selection, but also a performance benefit compared to a non-penalized model.

In other cases, there will be no performance benefit from penalization. The driving factors are the complexity of the model and the size of the dataset. As the size of the dataset increases, more complex unpenalized models can be fit with limited risk of overfitting. While we can describe this tradeoff in general, we cannot know without checking whether overfitting is happening, and therefore using average cross-validation performance (or similar methods) to check is required.

At each level of penalization, the parameters fit are different. Understanding how the parameters change as penalization increases can be useful, and we discuss this further in Chapter 7, Section 7.4.4.

Above, based on Figure 4.6, we selected the model with the best cross-validation performance and with 57 nonzero parameters. We can see that performance is quite flat over a wide range of penalty sizes, and an even simpler model could have been chosen for barely any performance reduction. If model simplicity is important to us, we might accept a small performance degradation in exchange for a less complex model. We should also remember that average cross-validation error (calculated by taking the mean across all the folds) is only an estimate of the generalization error—we don't know the exact value. Hence, the true generalization error of what we took to be the best model might in any case be less than the maximum cross-validation performance. We discuss this further in Section 4.6.5.

4.5. The Relaxed LASSO

In our case study above, the LASSO had two effects. It removed some features completely from the model. For example, it set the coefficient for type_registrant9 (Non-Citizen Co-Owned) to exactly zero. In addition, for all other features left in the model, their fitted parameters were smaller in absolute value than they would have been in a non-penalized regression (using the same nonzero features). For example, the linear predictor for aircraft age is shrunk toward zero; from −0.791 in the unpenalized model to −0.731 in the LASSO.

In general, when using the LASSO, each feature will either be set to exactly zero or will be reduced in absolute value by an amount regulated by λ (see Section 4.7.3 for a proof of this in a simple case). This effect is known as soft-thresholding. An alternative to soft-thresholding is hard-thresholding, where, if the coefficients are not set to zero, they are left at their original values. Figure 4.8 illustrates the difference.

To illustrate an extreme example, we show in Table 4.4 the coefficients in our FAA-NTSB model for a fairly high value of λ, only five out of 105 features have been selected. The table also shows coefficients from fitting an unpenalized model using only the five chosen features, and the final column shows the difference between the penalized and unpenalized coefficients (i.e., the shrinkage).

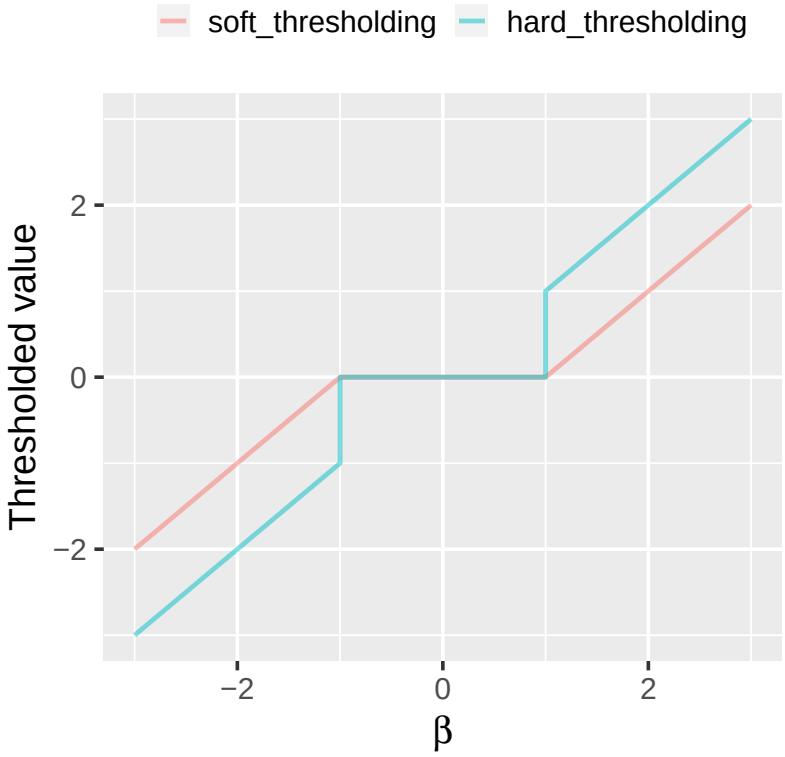

Figure 4.8. Soft-thresholding compared to hard-thresholding. In hard-thresholding, the coefficient drops suddenly to zero. In soft-thresholding, when the coefficient is not zero, it is still shrunk towards zero so the transition is smoother.

Table 4.4. For a fairly high value of λ, only five features are selected. The LASSO column shows the shrunken coefficients. The unpenalized column shows the fitted coefficients, where the selected features are fitted in an unpenalized model; the unpenalized values are significantly higher. There is no reason to assume that the shrunk coefficients will provide the best performing model for a model with only these five features.

	coefficients		
var	LASSO (a)	unpenalized (b)	shrinkage (b − a)
type_registrant3	0.320	0.545	0.225
region2	−0.070	−1.194	1.124
regionC	−0.093	−1.138	1.045
faa_eng_type11	0.299	3.716	3.417
veh_age	−0.003	−0.071	0.069

If we choose to use this simple model—with only five features—there is no reason to assume that the best validation performance is with the amount of shrinkage shown. And, in general, there is no guarantee that the penalized model which selects the best subset of features will also fit coefficients for the selected features which are best for model performance. In terms of the bias-variance tradeoff, once the LASSO has selected the features to include, the continued penalization of the remaining coefficients might be adding extra bias without reducing the variance.

One alternative is to allow those features whose coefficients are shrunk to exactly zero to be removed from the model but choosing the unpenalized coefficients for the other features. This is an example of hard-thresholding discussed above.

The relaxed LASSO (Meinshausen (2007)) allows for the above approach. It splits model training into two separate parts. In the first stage, feature selection is achieved. In the second stage, shrinkage is considered. It allows for hard-thresholding (i.e., no shrinkage in the second stage), but it also allows for some shrinkage to be chosen. Running the relaxed LASSO and comparing the CV performance curves allows the practitioner to benefit from feature selection while still ensuring optimal model performance.

4.6. Practically Speaking

4.6.1. Software

A pricing department, which has so far been limited to unpenalized regression and wishes now to use penalized regression, will need to choose a software. We used one or two standard open-source software[17] to carry out the analyses in this monograph. However, there are many open-source and proprietary software tools available for fitting penalized regression and other types of model, and the choice is not simple. Factors to consider go far beyond the ability of the software to actually fit a model and are beyond the scope of this monograph.

4.6.2. How many models?

Software used to train penalized regression models will often fit models for a sequence of λs from high to low, in order to be able cover in detail the full range of models from simple intercept-only models to models with (almost) no penalization. Typically, models for 100 values of λ may be fitted. If we are using cross-validation with 10 folds, then 100 models will be fitted for each fold, and in addition 100 models for the full path of λs will be fit on the complete training data from which the final model will be chosen. Hence 1,100 models will be fitted. This is not as bad as it sounds because given a sequence of λs, the models are fitted in the order of most penalized first. This means that, mostly, for any

[17] R and Python.

model in the sequence, the coefficient for a feature will be just slightly bigger than it was in the previous model in the sequence. Therefore, the initial estimate of the coefficients for a given λ can start from the final estimated coefficients of the previous λ in the sequence. These are called "warm starts." Friedman et al. (2010) mention that there are actually examples where it was faster to compute the whole path starting with the largest value λ down to a small value of λ than the solution only at that small value of λ. In addition, the models for the different folds can be done in parallel on different CPU cores. Overall, the speed of finding the coefficients for all the models will depend on the hardware, how well the software uses the hardware, and also the various technical tricks that the algorithm uses to reduce the memory footprint and speed up the calculations.

For this exercise, our training dataset had about 2 million observations, and the open-source software we were using took approximately 40 minutes to fit all the models. On the same hardware, an alternative open-source software took many hours. Hence, if using open-source software, it is important to experiment with the different choices available.

4.6.3. Penalization parameter names—λ and γ

Amongst different software and academic papers, there is not always consistency in the Greek letter used to represent the penalization parameter. Common choices are λ or γ. In the software we used, the penalization parameter is represented by λ, and the elastic net mixing parameter is represented by α. We have done the same here. When using a new software, it is best to check the documentation rather than making assumptions.

4.6.4. Performance graphs—x-axis

The x-axis of our performance graphs displays $\log(\lambda)$. When there is little or no penalization, λ approaches zero, making $\log(\lambda)$ negative. Consequently, models with minimal or no penalization—which are the more complex models—appear on the left side of the graph, while simpler models appear on the right.

4.6.5. Choosing λ

We discussed above that in Figure 4.6, performance is quite flat over a wide range of penalty sizes and that we might accept a small performance degradation in exchange for a less complex model.

Generalization error is the expected performance of a model trained on our training data over future data that it has not yet seen. In all of our work, we use the mean cross-validation error as an estimate of the generalization error. Whenever we use a sample mean as an estimate, we can measure the uncertainty of this estimate by its standard error. In our work, the mean is over the seven partitions we created. For any value of λ, we can calculate the standard deviation (s.d.) of the seven values of pseudo-R^2 and

hence the standard error of the sample mean ($\frac{s.d.}{\sqrt{7}}$). In an early work on decision trees Breiman et al. (1984), there is a proposal that we should take the simplest model whose performance is within one standard error of the model with the best cross-validation performance. This approach was included as default in one of the early penalized regression software, and we demonstrate it in Figure 4.9. Each red dot is the average cross-validation performance for a given amount of penalization. The error bars show 1 standard error around these means. The best cross-validation performance less 1 s.e. (standard error) is 0.142, and a dashed horizontal line has been drawn at this level. Finally, the vertical line labeled "1 s.e." is at the simplest model whose cross-validation performance is 0.142 or more. Breiman et al. (Section 3.4.3) say that in a way, this model "has an accuracy comparable to" the model with the maximum cross-validation performance.

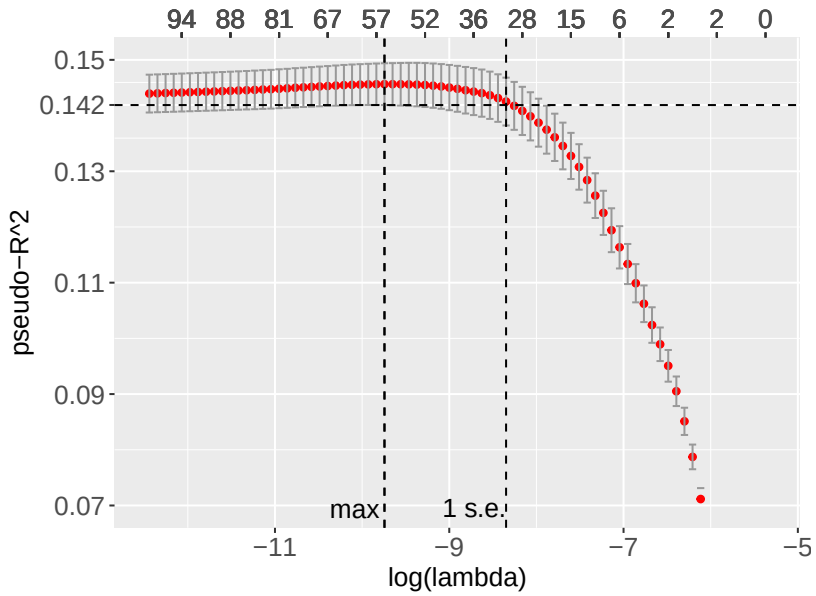

Figure 4.9. Choosing a model with a cross-validation performance within 1 s.e. of the best cross-validation performance. The error bars show 1 s.e. around the mean cross-validation for each level of penalization. The dashed horizontal line at 0.142 is at 1 s.e. below the best cross-validation performance. The simplest model whose performance is at or above this level is at the dashed line labeled "1 s.e."

The above approach should not be used mindlessly (and Breiman et al. certainly did not). It is introduced only to show the thought process which is available to us now that we have available a trade-off between performance and complexity. We can get further insight into this trade-off by comparing the average cross-validation error (which has been our focus until now) with the average training error. Figure 4.10 shows that for simpler

models (to the right of the x-axis) training and cross-validation performance are the same. As models start to become more complex, training performance increases slightly more quickly than cross-validation performance as models begin to pick up some random elements of the training data. At this stage, fitted parameters start to reflect effects in the training data which are not present in the validation data. At the "1 s.e." model the gap is quite small. At the "max" model the gap has widened; based on cross-validation performance, the fitted parameters still predict the better overall. However, it is possible that based on other unseen data, the extra noise picked up might reduce performance. This is a further incentive to choose a simpler model than the "max" model.

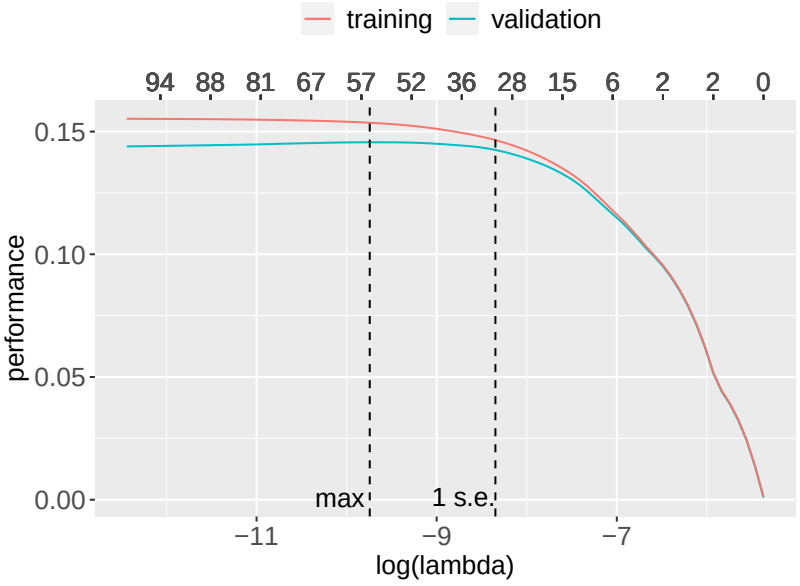

Figure 4.10. At the max model the validation curve is flat and the gap between the training and validation curve has widened. The models around this level of complexity have started to pick up patterns which are in the training data only and which do not improve cross-validation performance.

In our case, a further point to note is the consistency of performance differences for the individual folds. We do not show details here, but for every fold, performance of the "1 s.e." model is worse than the "max" model.

In practice, while good performance of insurance models is exceedingly important, simpler models are easier to understand and explain to underwriters, regulators, and policyholders. One option is to choose a model simpler than the "max" model, but still in the relatively flat part of the cross-validation performance curve and not as simple as the "1 s.e." model.

4.6.6. Flavors of penalized regression

When we train GLMs, the coefficients can take on infinitely many values. We hope that penalized regression helps us to choose a set of coefficients which will perform well in the future, but there is no guarantee that it will choose the best such model. In fact, there is no guarantee that penalized regression will even choose the model with the best average cross-validation performance.

The earliest type of penalized regression was Ridge regression. This gives a sequence of 100 (say) GLMs. Back when this was the only widely known type of penalty, the practitioner might have been forgiven for thinking that one of the 100 GLMs would be the best model. However, given that in our discussion above we have seen three different types of penalty and the fused LASSO, it should be clear that the sequence of models from any one approach is not guaranteed to contain the best model.

Another flavor of penalized regression is the adaptive LASSO (Zou (2006)). It deals with the problem in the LASSO that the strength of the penalty (λ) is the same for each feature, yet in practice this might not be reasonable. For example, we might, a priori, have a belief that some features should be in the model and that there is sufficient data to fit them correctly without any shrinkage. We might wish to ensure that such features are subject to much less penalization or are not penalized at all. The adaptive LASSO allows us to apply differing levels of penalty to each feature, depending on an initial estimate of their importance. We will see in Section 4.7.2 that after an adjustment called standardization, the coefficients from penalized regression model do in some way reflect the importance of the features. Hence the approach of the adaptive LASSO is to use the coefficients from a Ridge regression model to inform the amount of penalization applied in the adaptive LASSO.

Sometimes, data contain categorical features that measure the same thing in slightly different ways. This can arise, for example, where data have been enriched from different sources covering similar types of features. In such cases, model interpretation and implementation will be easier if levels are included from one categorical feature or the other, but not from both. A type of penalization, called the grouped LASSO, can help in this case. We do not go into further detail here.

Where does this leave the practitioner? While we cannot assume that Ridge regression or the LASSO will give the best possible models, they are certainly a very good starting point to help with both feature selection and model performance. Should there still be performance or feature selection issues, it is good to know that other flavors of penalization exist.

4.7. Technical Note

4.7.1. Solving Ridge regression

Consider the very simple case where there is only one feature (x) in our model. Assuming a normal distribution and identity link, unpenalized regression needs to minimize

$$l(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta x_i)^2.$$

This is easy to solve directly. The derivative is

$$\frac{\delta l}{\delta \beta} = \frac{1}{2n}\sum_{i=1}^{n} 2(y_i - \beta x_i) \times (-x_i) = -\frac{1}{n}\sum_{i=1}^{n} x_i y_i + \frac{1}{n}\beta \sum_{i=1}^{n} x_i^2,$$

and this is zero when

$$\frac{1}{n}\sum_{i=1}^{n} x_i y_i = \frac{1}{n}\beta \sum_{i=1}^{n} x_i^2.$$

So, the value of β which minimizes the loss function is

$$\hat{\beta} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i}{\frac{1}{n}\sum_{i=1}^{n} x_i^2}.$$

We now repeat this exercise with the Ridge regression penalty. Our loss function is

$$l(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta x_i)^2 + \lambda\frac{1}{2}\beta^2.$$

The derivative is

$$\frac{\delta l}{\delta \beta} = -\frac{1}{n}\sum_{i=1}^{n} x_i y_i + \frac{1}{n}\beta \sum_{i=1}^{n} x_i^2 + \lambda\beta = -\frac{1}{n}\sum_{i=1}^{n} x_i y_i + \beta(\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \lambda),$$

and this is zero when

$$\frac{1}{n}\sum_{i=1}^{n} x_i y_i = \beta(\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \lambda).$$

So, the value of β which minimizes the loss function is

$$\hat{\beta} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i}{\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \lambda}.$$

We can see that the impact of penalization is to shrink the estimate $\hat{\beta}$ towards zero (but not exactly zero) compared to the unpenalized estimate. The role of λ here is not totally dissimilar to the role of k in Bühlmann's credibility factor $Z = \frac{n}{n+k}$, which controls the extent to which group-specific estimates are shrunk towards the overall population estimate (see for example the discussion in Chapter 2 of Holmes and Casotto (2025)).

When there is more than one parameter, the relationship between λ and each of the fitted parameters is not as simple as that shown above. Each estimated parameter is still shrunk towards zero (but not exactly zero) compared to its unpenalized estimate, and the amount of shrinkage increases as λ increases.

4.7.2. Standardization

We saw in the last section that the parameter estimate under Ridge regression is

$$\hat{\beta} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i}{\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \lambda}.$$

Consider a case where frequency depends on length (in meters) and for which $\frac{1}{n}\sum_{i=1}^{n} x_i y_i = 10$ and $\frac{1}{n}\sum_{i=1}^{n} x_i^2 = 5$. The unpenalized estimate is $\hat{\beta} = 2$. The Ridge regression estimate will depend on the value of λ. If $\lambda = 5$, then $\hat{\beta}$ will be shrunk to 1. If we have an observation which is one meter long, the unpenalized estimate is a frequency of 2 and the penalized estimate is 1.

We now move to measuring length in centimeters. $\frac{1}{n}\sum_{i=1}^{n} x_i y_i$ will be 100 times bigger (1,000) and $\frac{1}{n}\sum_{i=1}^{n} x_i^2$ will be 100^2 times bigger (50,000). The unpenalized estimate is $\frac{1{,}000}{50{,}000} = 0.02$ and the penalized estimate is $\frac{1{,}000}{50{,}000+5} = 0.019$. The frequency estimates are 2 and 1.9. We can see that the impact of penalization changed just because we changed the units by which we measure the length.

If there is only one feature, the issue above does not matter—choosing an appropriate sequence of λs will still give a range of estimates from unpenalized down to zero. However, when there are many features, then at any level of λ, a feature which is measured in smaller units (and so has larger values and a smaller coefficient) will be less affected by the penalization. This can lead to some features being incorrectly penalized out of the model. The same issue applies in any penalized regression, not just Ridge regression.

In order to make penalized regression estimates independent of the units by which

features are measured, it is usual to standardize all the features before applying the rest of the calculation. This means deducting the mean and dividing by the standard deviation so that each feature has a mean of zero and variance of 1. The standard deviation is $\frac{1}{n}\sum(x_i - \bar{x})^2$. For a standardized feature, given that $\bar{x} = 0$, the standard deviation is $\frac{1}{n}\sum x_i^2 = 1$ so that parameter estimate under Ridge regression simplifies when features are standardized from

$$\hat{\beta} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i}{\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \lambda}$$

to

$$\hat{\beta} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i}{1 + \lambda}.$$

The default behavior of most penalized regression software is to standardize the numeric features before fitting but to return the coefficients which can be applied to the original features. It can be useful when comparing coefficients to do so on the scale of the standardized features. Table 4.5 shows the coefficients in our fitted model for the number of engines and aircraft age. On the original scale, the magnitude of the coefficient for the number of engines is bigger in absolute value than the coefficient for the number of engines. The number of engines ranges from one to four. The aircraft age ranges from 0 to 50, and despite its smaller coefficient, it will have a large effect on the linear predictor and it is more important in our fitted model. The coefficients on the scale of the standardized features are comparable, and we can see that indeed the magnitude of the aircraft age coefficient is larger.

Table 4.5. After fitting a penalized model on the standardized numeric features, the coefficients can be viewed on the original scale or on the scale of the standardized features.

var	original scale	standardized
faa_acft_num_eng	−0.277	−0.119
aircraft_age	−0.073	−0.791

Categorical features are typically represented by a column of zeros and ones. For example, for type of registrant category 3 (corporations), there will be a column which is 0 if the registrant is a corporation and 1 if it is not. Such features are often not standardized. Their values are all 0 or 1, and so they are not on very different scales. And their coefficients are easy to interpret. From a practical perspective, some software benefits in memory usage and speed from sparse data (lots of zeros) and standardization of these 0-1

variable would remove that benefit. If optimal model performance is key, then fitting a model both with and without standardization of categorical features and comparing the cross-validation performance curves is straightforward. In insurance pricing, a careful review, and manual adjustment if necessary, of the coefficients in the final rating plan is crucial. This is particularly true where adjustments to the model fitting process make a large difference to coefficients. Hence, if different approaches to standardization make a large difference to fitted coefficients, great care should be taken to review the model based on an understanding of the line of business.

4.7.3. Solving the LASSO

Reverting to our simple unpenalized case,

$$l(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta x_i)^2,$$

we saw that

$$\hat{\beta} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i}{\frac{1}{n}\sum_{i=1}^{n} x_i^2},$$

which, if x is standardized, is

$$\hat{\beta} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i.$$

Now consider the LASSO, we need to minimize

$$l(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta x_i)^2 + \lambda|\beta|.$$

Multiplying out the first term gives

$$\frac{1}{2}\left(\sum_{i=1}^{n} y_i^2 - 2\beta\sum_{i=1} x_i y_i + \beta^2\sum x_i^2\right).$$

If x is standardized, then this simplifies to

$$\frac{1}{2}\left(\sum_{i=1}^{n} y_i^2 - 2\beta\hat{\beta} + \beta^2\right),$$

where $\hat{\beta}$ is the unpenalized estimate.

Ignoring the constant $\sum_{i=1}^{n} y_i^2$, we need to figure out how to minimize

$$\frac{1}{2}\left(-2\beta\hat{\beta}+\beta^2\right)+\lambda|\beta|.$$

For convenience we add the constant $\hat{\beta}^2$ so that this becomes

$$\frac{1}{2}(\hat{\beta}-\beta)^2+\lambda|\beta|,$$

which given λ and $\hat{\beta}$ is a function of β and differs only by a constant from the original loss function $l(\beta)$.

Consider the case when the unpenalized estimate $\hat{\beta} > 0$. It can never make sense to set β to be more than $\hat{\beta}$ because the first term in the loss function ($\frac{1}{2}(\hat{\beta}-\beta)^2$) will be positive, and we can easily do better by reducing β to $\hat{\beta}$ which will set the first term to zero and at the same time reduce the second term. So the impact of the LASSO must always be to shrink the coefficients towards zero.

Now consider what happens when $\lambda \geq \hat{\beta}$. If we set $\beta = 0$, then the loss function is $\frac{1}{2}\hat{\beta}^2$. We will now show that any other value of β gives a higher value for the loss function, which implies that whenever $\lambda \geq \hat{\beta}$, β will be set to exactly zero.

If we allow β to be more than zero, then the loss is

$$\frac{1}{2}\left(\beta^2+\hat{\beta}^2-2\beta\hat{\beta}\right)+\lambda\beta.$$

Since $\lambda \geq \hat{\beta}$, we know that $\lambda\beta \geq \hat{\beta}\beta$ and so the loss is greater than

$$\frac{1}{2}\left(\beta^2+\hat{\beta}^2-2\beta\hat{\beta}\right)+\hat{\beta}\beta=\frac{1}{2}\left(\beta^2+\hat{\beta}^2-2\beta\hat{\beta}+2\beta\hat{\beta}\right)=\frac{1}{2}\left(\beta^2+\hat{\beta}^2\right),$$

which is certainly more than $\frac{1}{2}\beta^2$.

Therefore, for all values of λ which are greater than the unpenalized estimate $\hat{\beta}$, our penalized estimate is zero.

When $\lambda < \hat{\beta}$ our estimate of β can be greater than zero. When $\beta > 0$ the derivative of the loss function is

$$\frac{\delta l}{\delta \beta}=\beta-\hat{\beta}+\lambda.$$

At the minimum this is zero and we have

$$\beta=\hat{\beta}-\lambda.$$

Overall therefore, when the unpenalized coefficient $\hat{\beta}$ is greater than zero, the penalized estimate reduces that coefficient by the value of λ, until λ is greater than the unpenalized estimate and the penalized estimate is exactly zero.

The same simple relationship does not hold if there is more than one feature. In that case we can still say that given the final fitted penalized model, if we look at one coefficient given all the other fitted coefficients, it will be shrunk towards zero by some amount which will vary for each coefficient but still depends on λ.

When we move to using nonlinear link functions such as the log-link, the simple relationship breaks down further. However, it is still true that the parameters are either shrunk to exactly zero or reduced in size by an amount which gets bigger as λ gets bigger.

4.8. Next Steps

We have now fitted a penalized GLM on the FAA-NTSB data. The model includes aircraft age. Figure 4.11 shows the actual accident rate by aircraft age and the average prediction from the model. The frequency predicted by the model (blue line) is reasonably close to the actual frequency (red points), but in some places the shape is not quite right. For example, from aircraft age 30, the model predicted frequency continues to decrease—driven by the aircraft age effect in the model—whereas the actual frequency is reasonably flat.

The aircraft age effect, shown by the green line, is limited by our use of the GLM to being a very simple shape. The fitted model is

$$\log \text{frequency} = \cdots - 0.0824 \times \text{aircraft age},$$

or, taking exponents of both sides:

$$\text{frequency} = \cdots \times \exp\left[-0.0824 \times \text{aircraft age}\right].$$

This gives the smooth decreasing green line in the figure, but the actual frequency does not decrease in such a smooth manner.

In the next section, we will look at some approaches for modeling effects which have shapes not easily picked up by GLMs.

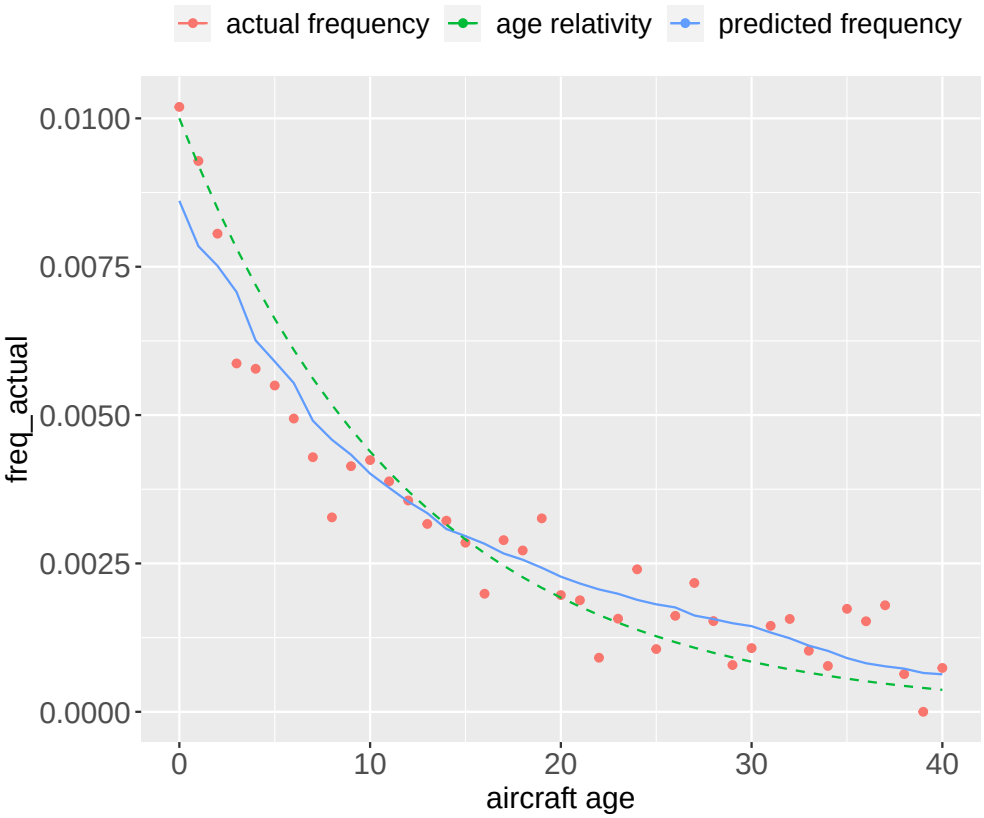

Figure 4.11. FAA-NTSB LASSO on features excluding HCCVs. The model predicted frequency (blue line) is close to the actual frequency (red points). However, from age 30 the predicted frequency continues to decrease, driven by the aircraft age effect in the model, whereas the actual frequency is reasonably flat.

5. Nonlinear Effects

5.1. Feature Engineering

The aim of our models is to capture the relationship between things we know about the risk and the frequency of accidents. We refer to the things we know about our risk as features. We have seen above that, given a set of features, the LASSO and elastic net can select the relevant features based on your data. This is known as "feature selection."

GLMs cannot create features themselves. For example, we might know that the weather at or near certain locations is extreme, and therefore certain types of claim have a higher frequency at or near those locations. We know that for each policy we need to calculate the distance from the nearest extreme weather area and to use that as a feature. Now we add to each line of our data the location of all the extreme weather areas and of the policy itself. Although we have put the information into the dataset, the GLM will not do the calculation itself and will not pick up the effect. For example, Table 5.1 shows some sample data. Each observation contains the location of a risk and the location of high risk weather area 1. The GLM will not pick up the fact that location A is near high-risk weather area 1 and location B is not.

Table 5.1. Even if all of Latitude, Longitude, Latitude_1, and Longitude_1 are features, a GLM will not pick up that location A is near high-risk weather area area 1 and location B is not. We need to add a new feature, distance_1, which is the distance from the high-risk weather area.

Location	Latitude	Longitude	Latitude_1	Longitude_1
A (Metairie, Louisiana)	29.9841° N	90.1529° W	29.9511° N	90.0715° W
B (Sacramento, California)	38.5816° N	121.4944° W	29.9511° N	90.0715° W

We need to do the calculation and add just one new column—the distance from the nearest extreme weather area—into the dataset. In our example, we add a new feature, distance_1, which is the distance of each location from high-risk weather area 1 (9 km for location A, and 3,000 km for location B). Creating a new feature is called feature engineering.

Another example of feature engineering is the creation of a credit score. It might be that the credit score is created from 50 features. While we could include all 50 features in the model, there are various reasons why we might want to create a credit score feature and use only that:

- The model output will be easier to interpret.
- If many of the features underlying the credit score are only marginally effective, penalized regression might remove them completely from the model so that the final credit score implied by the fitted model is not as strong as it could have been.
- Credit score as a whole might strongly interact with other features in the model, whereas this interaction would be very hard to fit to the individual components of the credit score even if it is visible.
- There may be a separate department (or an external vendor) which understands all credit-related features (and the legal aspects of which can be used and which cannot). They might best be placed to build and maintain a credit score model, passing only the result to the insurance model-building team.

Sometimes it is not obvious that we need to engineer new features. Consider aircraft age on our FAA-NTSB model. We know that accident frequency varies with aircraft age, and we have the aircraft age feature on our model. However, as we have seen above, this is not sufficient because the relationship is not linear in aircraft age (or the log of aircraft age). We need to somehow engineer new features to allow our GLM to pick up the right shape for this effect.

Goldburd et al. (2025) discuss three approaches:

- binning the variable
- adding polynomial terms
- using piecewise linear functions.

They note various disadvantages of the first two approaches. Binning the variables can add many extra parameters into the model, can lead to lack of continuity in the estimates, and will ignore variation within the bins. Polynomial terms may behave erratically at the edges of the data.

We will look at two types of piecewise linear functions: step functions and hinge functions. After discussing each of these, our approach will be simple. We will create lots of them, add them to our data, and then use LASSO or elastic net to choose those most relevant to our model based on our data.

5.2. Step Functions

A step function is a function which outputs 1 if its input is at or above a threshold and zero otherwise. Table 5.2 shows an example. The function takes aircraft_age as an input and returns 1 when aircraft age ≥ 30 and zero otherwise.

Table 5.2. A step function which has a threshold at aircraft age 30.

aircraft_age	aircraft_age_stp_30
1	0
7	0
17	0
31	1
42	1

Let's remove aircraft age from our model and replace it with the step function in Table 5.2—the first column will be replaced by the second column.

We now fit a model using only the feature aircraft_age_stp_30. The fitted model is

$$\log \text{frequency} = -5.2815 - 1.4689 \times \text{aircraft\_age\_stp\_30}$$

or, taking exponents of both sides

$$\text{frequency} = 0.00509 \times \exp[-1.4689 \times \text{aircraft\_age\_stp\_30}].$$

In other words, the model predicts a frequency of 0.00509 for aircraft ages 29 or less and 0.00509×0.2302 for aircraft ages 30 or more ($exp[-1.4689] = 0.2302$). This result is shown in Figure 5.1.

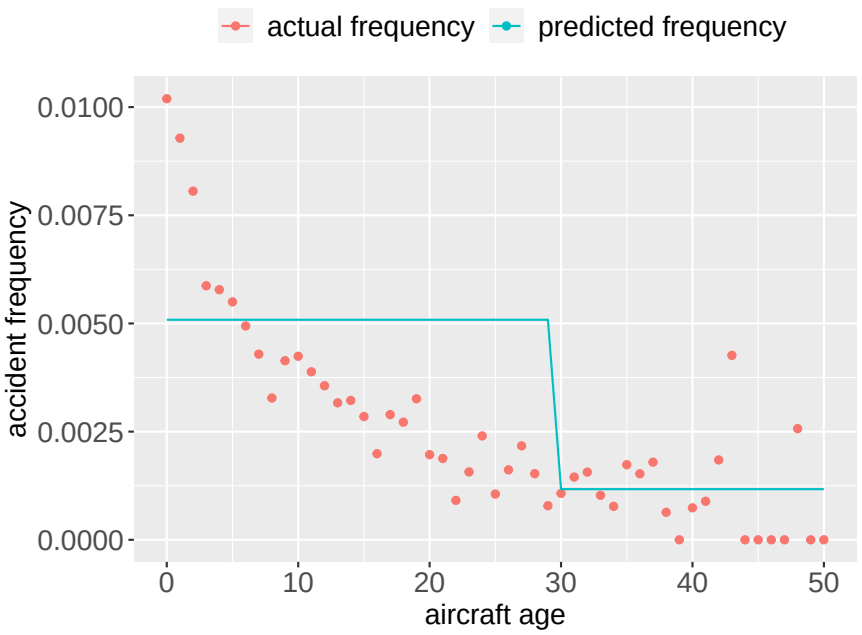

Figure 5.1. A model for aircraft age using one step function only. The step is at age 30. The GLM picks up the lower frequency after age 29 but has not been given any features to be able to pick up any changes in frequency between age 0 and 29.

We can see visually that there are some big frequency changes near aircraft ages 5 and 15 which the GLM could not model since it was not given the relevant features. We can therefore add two more features—these being step functions at 5 and 15. Our model will now use the columns shown in Table 5.3 (except aircraft age).

Table 5.3. Step function with thresholds at aircraft ages 5, 15 and 30.

aircraft_age	aircraft_age_stp_5	aircraft_age_stp_15	aircraft_age_stp_30
1	0	0	0
7	1	0	0
17	1	1	0
31	1	1	1
42	1	1	1

The fitted model is

$$\begin{aligned}\log \text{frequency} = &-4.8292 \\ &- 0.6881 \times \text{aircraft\_age\_stp\_5} \\ &- 0.5711 \times \text{aircraft\_age\_stp\_15} \\ &- 0.6237 \times \text{aircraft\_age\_stp\_30}.\end{aligned}$$

Taken exponents, this simplifies to

$$\text{frequency} = \begin{cases} 0.00783 & \text{if aircraft age} < 5 \\ 0.00394 & \text{if } 5 \le \text{aircraft age} < 15 \\ 0.00222 & \text{if } 15 \le \text{aircraft age} < 30 \\ 0.00119 & \text{if aircraft age} \ge 30. \end{cases}$$

This result is shown in Figure 5.2. The shape of the GLM predicted frequencies is closer to the actual frequencies but is still limited by the model only being given three features.

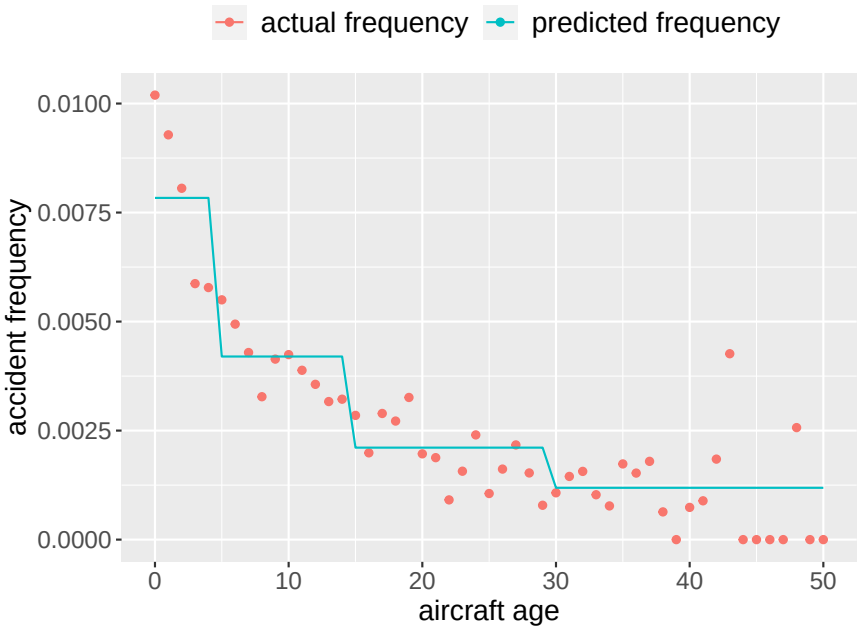

Figure 5.2. A model for aircraft age using three step functions only. The GLM is now able to pick up some of the key aspects of the shape, but there is clearly some detail still missing.

There is a problem with this approach so far—we are continually having to look at the result and decide ourselves whether or not it is good enough. Given that we know that LASSO and elastic net can simply select the features needed based on the data, the obvious solution is to give the GLM many features—one step function for each age and to allow the penalization to select the features which optimize k-fold cross-validation performance. We therefore create 50 features—one step function for each age and run a penalized GLM. The k-fold cross-validation curve is shown in Figure 5.3. We can see that at the best cross-validation performance there are 19 nonzero coefficients and that the simplest model with a cross-validation performance within 1 s.e. of this performance has 11 parameters. As we have previously seen, when the standard errors are large, the "1 s.e." model is not on the flat part of the cross-validation performance curve. Also, although not shown here, for every fold the performance of the "1 s.e. model" is not as good as the model with the maximum average cross-validation performance. We therefore choose the simplest model with a cross-validation performance within 2%[18] of the best cross-validation performance—the upper horizontal dashed line shows this level. Our chosen model has 17 step functions.

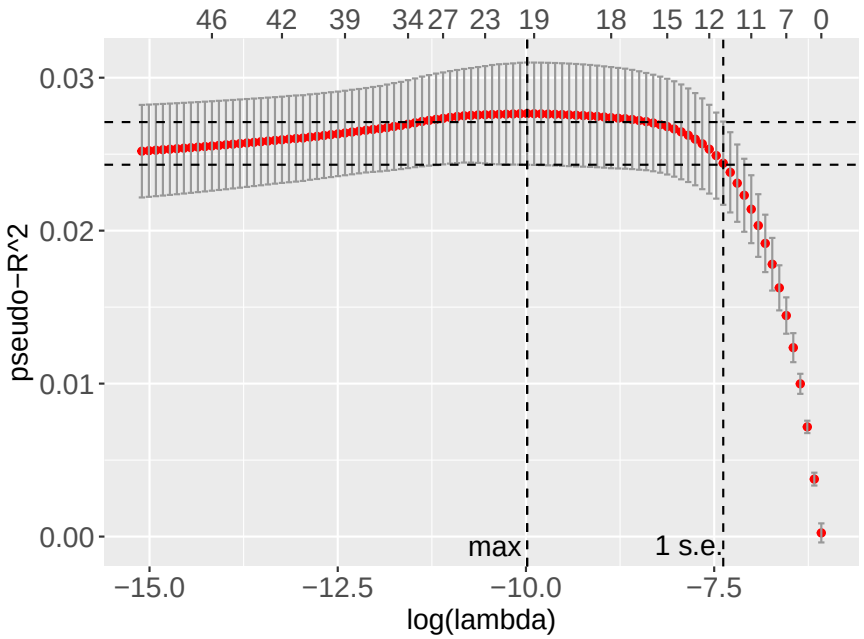

Figure 5.3. 50 step functions as features—one for each age. At the maximum validation performance, 19 of these features are selected. The "1 s.e." model is simpler, with only 11 features, but is not on the flat part of the cross-validation performance curve. The upper dashed line is at 98% of the best cross-validation performance, and we have chosen the simplest model with this performance.

[18] This is based on our judgment that the cross-validation performance is fairly flat at this point, and we prefer a slightly simpler model where possible, as it might generalize better.

Figure 5.4 shows the model predictions compared to the actual frequency. The penalized GLM has come close to picking up most of the shape of how actual frequency varies by age but without overreacting to the noise. However, at aircraft ages greater than 20 years old the predictions look too high. The reason for this is that the LASSO has not only selected the best step functions, it has also shrunk the coefficients of the remaining ones—and this is not necessarily optimal. A solution (the Relaxed Lasso) is discussed in Section 4.5.

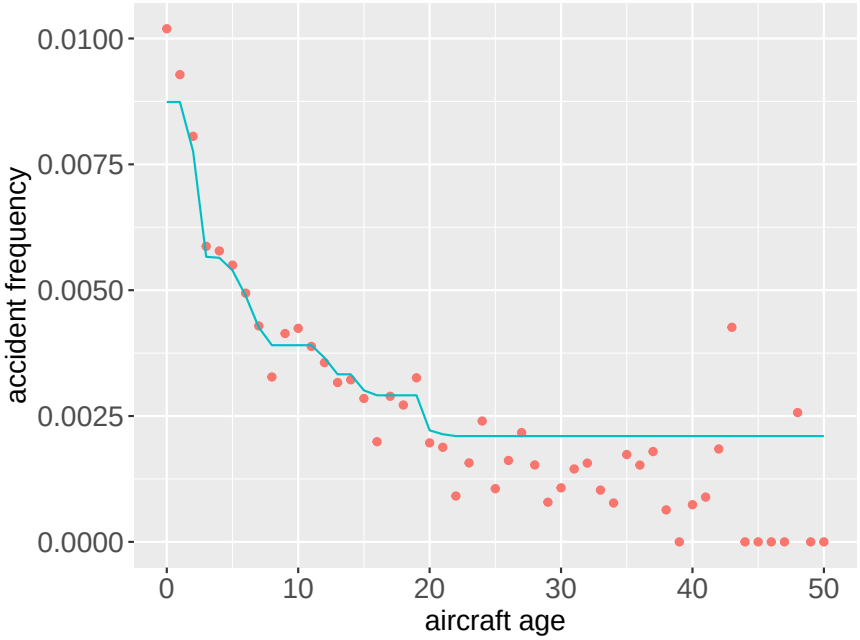

Figure 5.4. Predicted compared to actual frequency. The 17 step-functions selected pick up much of the shape of the actual frequency, though the predictions above age 20 look high.

5.3. Hinge Functions

If we look again at Figure 5.4 we see that the shape of the aircraft age effect is really made up of three parts: 0 to 3 years—a steeply reducing linear trend; 4 to 28 years—a less steep, but still reducing, linear segment; 29 years onward—a mostly flat segment. The flat step functions we have been using so far cannot easily pick up an effect which is a slope. We have to use lots of small flat line segments to pick up a slope. That is why we needed 11–17 step functions to get close to the correct shape. A simple way to pick up a shape which is made up of segments of straight-line slopes is to use "hinge functions." These are discussed in Goldburd et al. (2025), Section 5.4.4. Consider:

$$h(\text{aircraft age}, 28) = \max(0, \text{aircraft age} - 28)^{19}$$

Examples of some values of this function are shown in Table 5.4, and the shape of this function is shown graphically in Figure 5.5.

Table 5.4. A hinge function with a "hinge" at aircraft age 28.

aircraft_age	aircraft_age_h_28
0	0
20	0
28	0
29	1
49	21

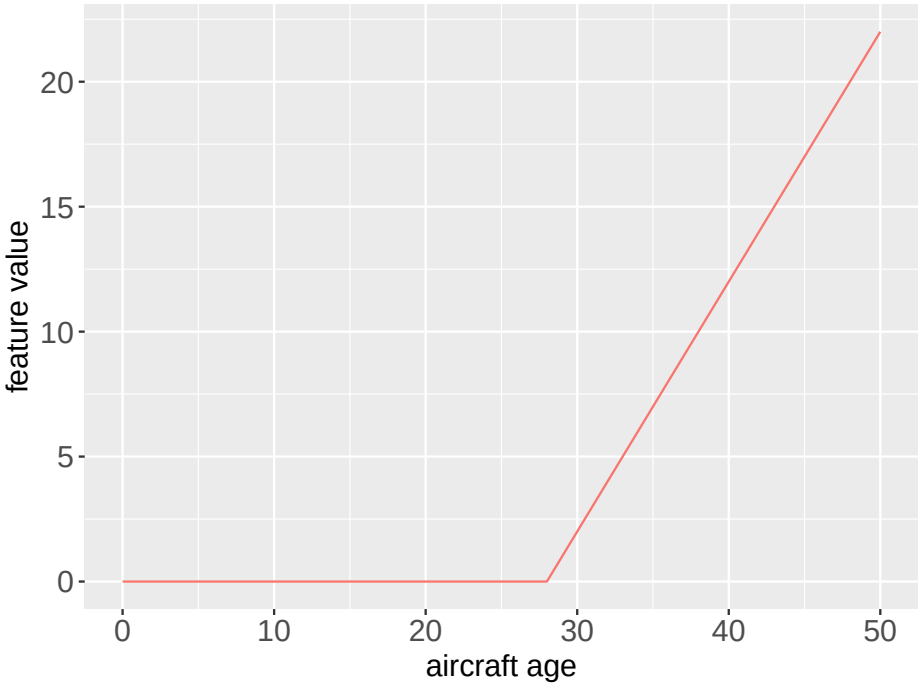

Figure 5.5. An example of a hinge function with a "hinge" at age 28. The function is zero until age 28 and then increases with age. If there is a change of slope of the age effect at age 28, this hinge function can pick it up.

[19] This is sometimes expressed as (aircraftage - 28)+

We now create two hinge functions so that our GLM will have (besides the intercept) three features, age itself (which is a hinge function with the knot at zero), and hinge functions with knots at 4 and 28. The penalized GLM fits the following model:

The fitted model is

$$\begin{aligned}\log \text{frequency} = &-4.5566\\ &-0.1599 \times \text{aircraft\_age}\\ &+0.1011 \times \text{aircraft\_age\_h\_4}\\ &+0.0336 \times \text{aircraft\_age\_h\_28}.\end{aligned}$$

We can see that the slope gets less steep for ages 4 and above and then again for ages 28 and above. The actual and predicted frequencies are shown in Figure 5.6. With only three features, the GLM has closely picked up the shape that we see in the data. This is not surprising, given that we choose the knots having seen the data. Besides being time-consuming if we were to do this for every continuous dependent variable, it is also suboptimal as we may be misled by patterns that we think we see in the data which are actually just noise.

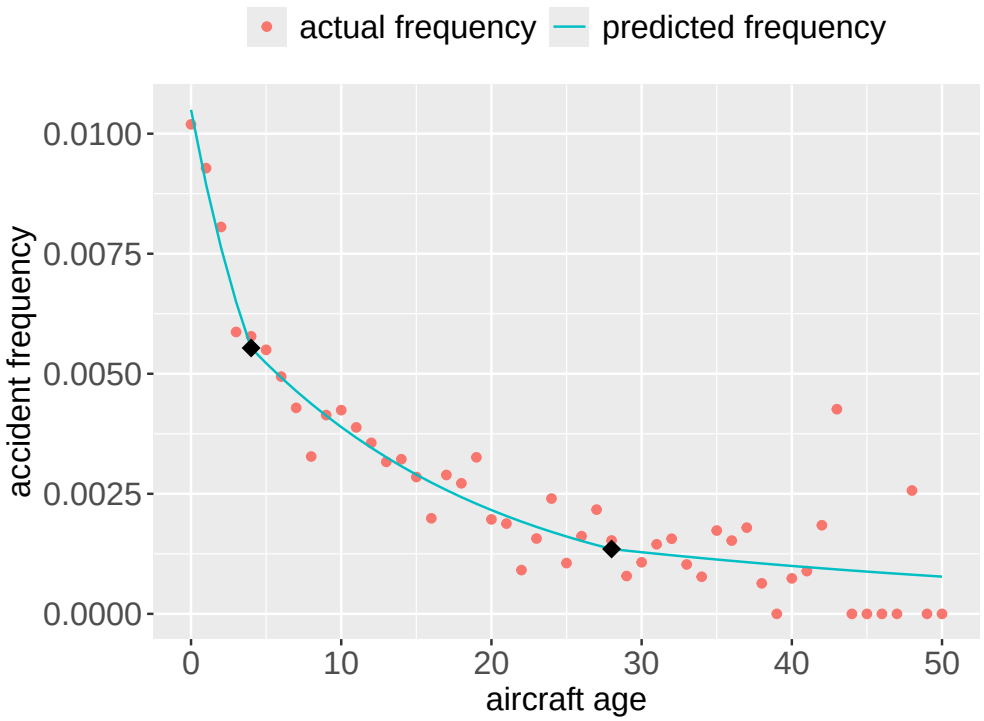

Figure 5.6. FAA-NTSB: A model for aircraft age using two hinge functions only. This GLM closely picks up the shape of the effect that we see.

As previously, one approach is to give the GLM 50 features—a hinge function with knots at each age. The k-fold cross-validation curve is shown in Figure 5.7. We can see that the performance is very similar for models for the maximum number of features down to much simpler models. As previously, due to the large standard error of the cross-validation means, the "1 s.e. rule" seems to pick a model which is too simple. We instead choose a model which is within 2%[20] of the best performance (the upper dashed line in the figure).

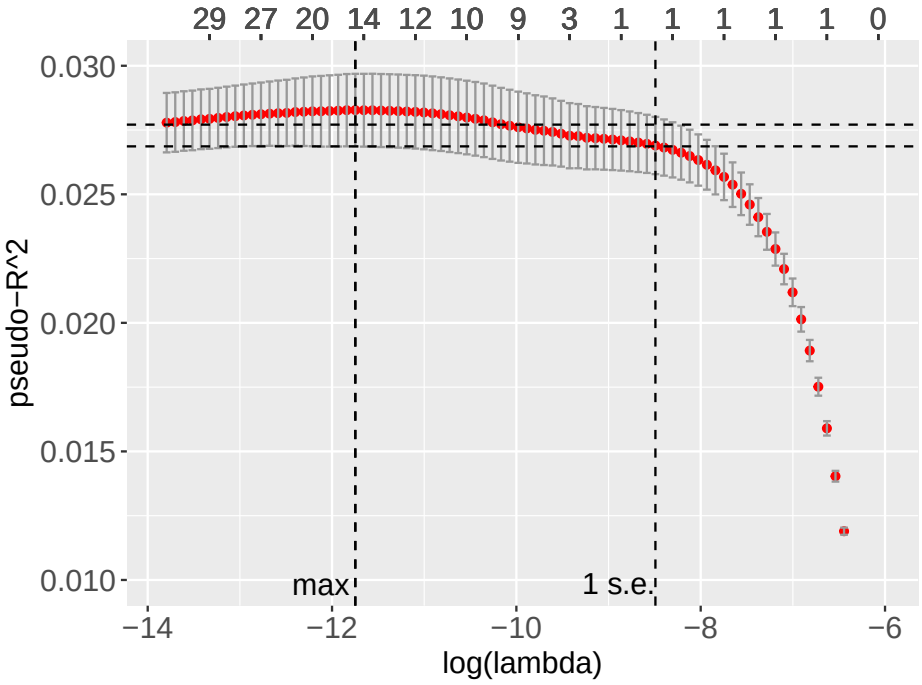

Figure 5.7. A model for aircraft age, using 50 hinge functions as features—one for each age. The left vertical dashed line shows the choice of log(λ) which maximizes pseudo-R^2. The average cross-validation performance is quite flat at that point. We choose instead a simpler model which has a performance within 2% of the maximum—the upper dashed line. That model has aircraft age and seven hinges.

The model chosen above uses only the original aircraft_age features and seven hinge functions. The resulting predictions are shown in Figure 5.8. They seem mostly reasonable, though predictions at ages 0 and 1 might be too low.

[20] As above, this is based on our judgment. It leads to a reasonably performing model being chosen, and at this point in our work we preferred simpler models, as they are easier to explain. However, a 2% performance reduction is significant, and later on, when we were looking to maximize performance, we did use models at the maximum of the cross-validation curve.

As an example of hard-thresholding (see Section 4.5), in Figure 5.9 we show a model with the same hinge functions as above, but without penalization.

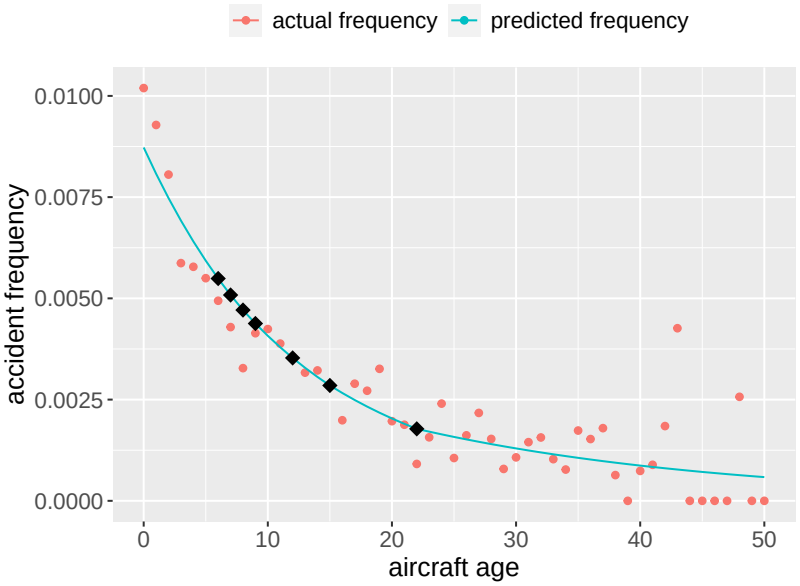

Figure 5.8. Predictions from a model using aircraft age itself and seven hinges (the black diamonds show the points chosen for the hinges). The predictions at older ages look better, but predictions for ages 0 and 1 may be low.

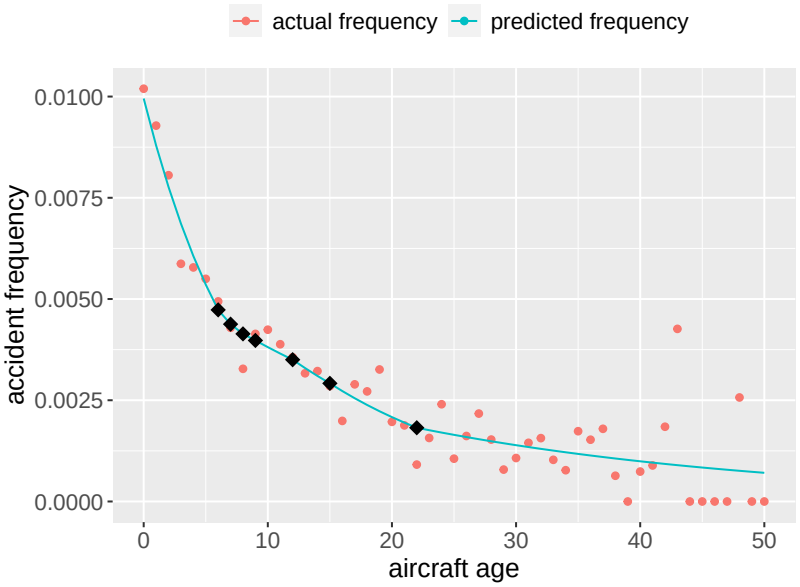

Figure 5.9. As an example of hard-thresholding. The age feature itself and the seven hinge functions chosen by the LASSO are used in a model with no penalization. The black dots show the points chosen for the hinges. Predictions for ages 0 and 1 are closer to the data.

5.4. Step Versus Hinge

Given our discussion of step and hinge functions, which is better? There is no clear answer, but the following points might be considered by the practitioner.

We mentioned previously that a disadvantage of polynomial terms is that they may behave erratically at the edges of the data. Step and hinge functions tend to behave in a more stable—but different—way at the edges. Step functions are flat at the edges of the data, and hinges may slope up or down.

Consider a feature such as annual mileage in private auto, where the last step or hinge function is at 10,000 miles, but some policies have a declared annual mileage of 100,000 or more. If we use step functions and the last step is 10,000 miles, then the price will be the same for all mileages greater than 10,000. If the last hinge was upward sloping at 10,000 miles, then prices at much higher mileage will be exceedingly high. The practitioner may prefer one approach or the other.

Sometimes, we want and expect the "customer journey" through successive renewals (i.e., change in premium charged) to be smooth. For example, we might wish the private auto price to go down smoothly as customer age increases from age 17 to age 30. A step function might be flat from age 17 to 20, followed by a significant drop at age 21, and then flat again from age 21 to age 30. While a practitioner can smooth the results of a GLM based on step functions, the cross-validation performance will no longer be available (unless the practitioner repeats the smoothing for each fold looking only at the training data). In these cases, hinge functions may be preferable.

There is one situation where monotonic step functions may be particularly useful: ordinal categorical features.[21] We consider this in more detail in Section 5.7.2.

If the practitioner does expect the price to suddenly change from one value of a feature to another, in a discontinuous manner, then step functions are preferred for that feature.

As an aside, we note certain similarities to machine learning techniques. Step functions are the functions used by tree-based predictive models, albeit usually with more complicated conditions for assigning a one to a record. They are therefore the basic functions used in random forests and gradient boosting machines. Hinge functions are a generalization of functions widely used in deep learning called Rectified Linear Unit (ReLU) functions. These functions are not linear, but they are differentiable almost everywhere and allow parameters in neural networks to be found through a differentiation process called back-propagation.

[21] Ordinal categorical features are categorical features which have a meaningful order—this will be explained later in detail.

5.5. Multivariate Adaptive Regression Splines (MARS)

Our approach above was to create many features (either step functions or hinges) and then to allow penalized regression to choose the most suitable hinges. We suggested creating one feature for each age. Doing this for every continuous variable will create possibly hundreds or even thousands of features, and fitting a penalized regression model on that many features is computationally expensive. If the full design matrix is stored in memory, then the problem can be memory intensive as well. While proprietary software exists which uses custom solvers to solve this problem, we instead choose a limited number of knots. Although this is likely to produce a less powerful model, if the knots are chosen carefully, the reduction in performance may be limited. One approach is to choose knots based on percentiles or weighted percentiles of the data. Alternatively (and as mentioned in Goldburd et al. (2025) Section 10.4) we can use a model type called multivariate adaptive regression splines (MARS[22]).

MARS (Friedman (1991)) can be considered a type of decision tree, but using hinges[23] rather than step functions. Consider a decision tree using aircraft age. At the start of training, the first decision will be created by finding the "best" aircraft age at which to split aircraft into two groups, one with higher frequency and one with lower. By "best" we mean that if we create predictions in this way, the performance will be better than if any other split was made. It is exactly the same as using one step function in our discussion above where that one step function is chosen so that it maximizes performance on the training data. Likewise at the start of training, MARS will find the best knot so that if it creates a spline at that point, and finds the best coefficient to apply to that spline, performance will be maximized on the training data. Again, this is exactly the same as using one spline function in our discussion above, where that one spline function is chosen so that it maximizes performance.

After the first spline, MARS will continue to add splines in a stepwise manner (similarly to stepwise forward linear regression), so that each added spline improves the performance more than any other spline would have done. MARS is designed with interactions in mind, in the same way as decision trees; however, typical implementations allow the user to ask for interactions not to be fitted—which is what we will do.

---

[22] Friedman, the inventor of this method, could equally have called it "Adaptive Piecewise Linear Regression for Multivariate Data," but that would not have had the same catchy acronym. Because the term "MARS" is trademarked, the implementation in R is called EARTH. Its creator, Stephen Milborrow, said that the "backronym" for EARTH is Enhanced Adaptive Regression Through Hinges.

[23] The "S" in MARS is an abbreviation for "splines." A spline is a function that, in some sense, smoothly passes through a set of points. MARS actually uses linear splines, which are essentially identical to the hinges we have discussed. We will use the terms "hinge" and "spline" interchangeably.

Given that MARS continues to add splines based only on improvements to training error, without some adjustment, it would be guaranteed to overfit. This was understood in its original implementation and a "pruning" method was included which removed the splines most likely to lead to overfitting. However, in a manner consistent with our general approach in this monograph, we turn off the pruning in MARS and allow the LASSO to select the relevant splines.

There is a further issue we need to consider. In our first discussion above, our step functions and hinges were created without reference to our target variable, in an unsupervised manner. We then proceeded to use cross-validation to find the best level of penalization. Cross-validation works because it uses data which our model fitting process has not yet seen. When we create splines with MARS, we are looking at the target variable in order to decide where to put the knots. If we use all of our data to create the MARS splines, by the time we get to cross-validation, our model fitting process has already seen the values of the target variable on the validation folds and the chosen value of λ will be too low. We therefore take care during cross-validation to create the splines separately for each fold.

Although we do not consider it further here, in the same way that MARS can be used to create spline-based features for inclusion in the LASSO, decision trees can be used to create the thresholds for step functions.

5.6. Case Study—Piecewise Linear Functions

Finally, we return to our FAA-NTSB case study. We still exclude the High Cardinality Categorical Variables (HCCVs) but for numeric variables, we use an approach based on the above discussion. We use three approaches to choose the features to use in the LASSO:

- Start with 25 equally spaced step functions for each numeric feature.
- Start with 25 equally spaced hinge functions for each numeric feature.
- Allow MARS to select the starting hinges. In this case, the starting hinges are created separately for each fold.

Table 5.5 compares the cross-validation performance and the number of nonzero step or hinge functions chosen for three methods.

Table 5.5. Model performance (pseudo-R^2) and number of nonzero features based on three ways of choosing the starting features to capture nonlinear effects.

feature creation	cross-validation performance	nonzero features
step functions	0.150	67
hinges	0.149	77
MARS hinges	0.150	63

The three approaches have similar levels of performance. We don't have a preference between the step functions model and the MARS hinges model. Performance-wise, they both perform slightly better than the hinges model. The step functions model is easier to fit than the MARS model (we do not need to add the MARS step into the model building of each fold). The MARS hinges function uses slightly fewer features and will probably require less manual adjustment (interpolation) when creating the actual rate plan.

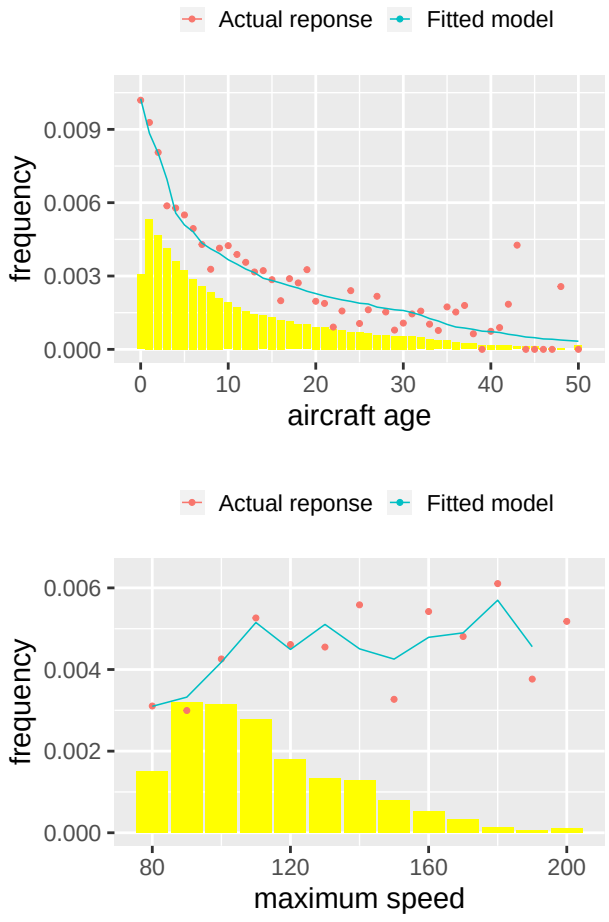

Figure 5.10. The model seems to pick up key aspects of how frequency varies with the features shown. The height of the yellow bars indicates the relative amount of exposure for each value on the x-axis.

One issue with the fitted model is that the predictions continue to reduce as the aircraft age increases—even at high ages, where there is little data. This is the result of extrapolation based on the hinges, as discussed in Section 5.4. Where a practitioner prefers hinge to step functions but is concerned by this issue, they could consider manually adjusting the values of the hinge functions to be constant at the extremes of the data (for example, for all values of each numeric feature greater than the 99th percentile).

5.7. Practically Speaking

5.7.1. Random features

In the above discussion on MARS, we allowed it to generate hinges which we then fed into the LASSO to benefit from its implicit feature selection. Depending on the values of certain MARS parameters, it will continue to generate hinges, even when those hinges are picking up random noise in the training data, rather than signal. While we rely on the LASSO to ignore irrelevant features, as datasets increase in size, having tens or possibly hundreds of irrelevant features can cause computation time and resource requirements to increase. We therefore added a random numeric feature[24] to our data. Once the MARS generated list of hinges started to select hinges based on this random feature, we knew that it must be picking up random noise (because there is no signal in the random feature), and we stopped the process from generating further hinges.

In general, the idea of adding one or two randomly generated features to a dataset can be very useful. If whichever process is used to create the model ends up including these features, we ought to be concerned.

The idea of "shadow" variables in some machine learning techniques is similar. Each feature is randomly shuffled and added back to the data alongside the original feature.[25] If, in the final model, the random version of the feature is more important than the original feature, there is probably something wrong with the modeling process.

5.7.2. Ordinal categorical variables

Consider a feature "quality of pilot" which takes the values poor, worse than average, average, better than average, and good. This is a categorical feature—it has five different categories. There is also meaning in the order of the variable—each level of the category is "better" than the previous level. Hence, this feature is called an ordinal categorical variable. In a GLM, one simple way we can deal with ordinal categorical variables is as follows. We first replace the text of the categories with numbers. "Poor" becomes 1,

[24] We simply created a new feature which was the row number of the dataset, and then we shuffled it.

[25] See for example Kursa and Rudnicki (2010). In our MARS work, too, a shadow variable could be created for each feature, and hinges for that feature would not be used after MARS starts creating hinges for its shadow feature.

"worse than average" becomes 2, and so on. We should not simply put this new numeric feature in the GLM. That is because although 2 is twice 1, it is not true to say that "worse than average" is twice "poor." Actually, even if that were true, there is no reason to assume that incident frequency will increase or decrease linearly with quality of pilot. However, given our discussion above, what does seem natural is to use step functions at each level of quality. Each level of pilot quality will be represented by five step functions (Table 5.6 shows their values).

Table 5.6. Values of step functions applied to levels of an ordinal categorical feature.

category	step_1	step_2	step_3	step_4	step_5
1 – good	1	0	0	0	0
2 – better than average	1	1	0	0	0
3 – average	1	1	1	0	0
4 – worse than average	1	1	1	1	0
5 – poor	1	1	1	1	1

If accident frequency for some level of quality and above is worse (or better) than accident frequency below that level of quality, then the step function will pick this up. For example, if a nonzero linear predictor is fitted for step_2, it will be applied to all pilots whose quality is less than good.

We don't have pilot quality on our FAA data, but to provide some sample output, we simulated this feature. Table 5.7 shows the fitted coefficients for each step function.

Table 5.7. The first row shows the coefficient for the step function. The next row shows the exponent of the coefficient. Finally, the last row shows for each level of the ordinal categorical feature, the relativity applied to a pilot with that level of experience.

category	step_1	step_2	step_3	step_4	step_5
coefficient	0	0.515	0	0.211	0.206
exponent of coefficient	1	1.673	1	1.235	1.229
cumulative product	1	1.673	1.673	2.06	2.539

Table 5.8 compares the fitted effects to the true relativities (which we know because we simulated the data). We can see that the step functions have correctly fitted an effect which increases for poorer quality and which is not linear.

Table 5.8. A comparison of the ground truth with the fitted effect. We can see that the fitted effect is monotonically increasing. The fitted effect for some levels of the factor is quite far out—this is driven by the randomness (and limited volume) of the simulated data.

category	true relativity	fitted effect
1 – good	1.00	1.00
2 – better than average	1.66	1.67
3 – average	1.80	1.67
4 – worse than average	2.01	2.07
5 – poor	2.06	2.54

We decided, before fitting this model, that we wanted the pilot quality effect to be monotonic increasing. For example, we wanted the relativity for a "worse than average" pilot to be more than an "average" pilot. We achieved this by adding a constraint on the linear predictors for each level of pilot quality to be non-negative—hence the coefficients in the first row of Table 5.7 were guaranteed to be non-negative. This ensured that the relativity applied to each function was greater than or equal to 1 and so the overall cumulative effect is monotonic increasing. Adding such constraints is often not a good idea. It is often better to "let the data do the talking."

If the effect of pilot quality on accident frequency is indeed monotonic increasing then, as long as we have sufficient data in each class, we will get that effect. On the other had if our classification process for quality of pilot is poor, so that the accident frequency for a "worse than average" pilot is actually better than that of an "average" pilot, the constraints will hide this effect and we will not be able to feed the problem back to the business or whoever created the classification system. Some practitioners therefore have a strong preference to avoid such constraints.

A further example of the above issue is the discount scale in private auto insurance for where a car is parked overnight. There are typically at least three levels for this feature; 1—garaged, 2—parked on a driveway, 3—parked on the road. We might have the expectation that the frequency of claims will be least for the garaged car. For example, it is least likely to be stolen or vandalized or knocked into overnight by a car driving carelessly down the road. It might be tempting therefore to treat it as an ordinal categorical feature and apply non-negative constraints to the linear predictors.

However, in practice, the level 1—garaged feature often has a higher claims frequency than the others. This is because the parked overnight feature is self-declared, and a significant proportion of vehicles declared as parked overnight in a garage are not (in fact, a fair percentage of such policyholders don't actually have a garage). Beyond misreporting

of this feature, it may also serve as an indicator for the potential misreporting of other self-declared information.

In some cases, insisting on monotone increasing effects does seem reasonable. For example, for the number of historic claims declared at policy inception. We should expect future claims frequency for a policy to increase in line with historic frequency. However, often there is only a small amount of data for three or more historic claims. This can lead to three or more claims being predicted with a lower frequency than one or two claims, which would not be a reasonable rating plan to bring to the market.

5.7.3. Surrogate GLMs

Various machine learning model types are much more flexible than GLMs and, with sufficient data and careful tuning, have the potential to perform better than well-fitted GLMs in predicting the value of a target variable given the features. In addition, they can just be given the numeric features and will find a way to make good predictions even if the effect is not linear with the numeric feature. A downside to some of these models is that they are not easy to interpret—there is no easy way to explain to a human the link between the value of the features and the prediction output by the models. Techniques have therefore been developed to help interpret the outputs of such models.

In *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*, Molnar (2020) provides an accessible introduction to this topic. One of the approaches he mentions is "surrogate models." This is explained as when "an interpretable model...is trained to approximate the predictions of a black-box model." A surrogate GLM is a GLM which is trained to approximate the predictions of a machine learning model which is not easily interpretable.

This suggests the following approach to nonlinear effects:

- Train a good machine learning model on the data (for example, a Gradient Boosting Machine)
- Extract information about the shape of the nonlinear effects
- Use the above information to fit a GLM

5.8. Technical Note

5.8.1. Step functions and the fused LASSO

The penalty term used in penalized regression will impact both the features in the fitted model and their coefficients. In our work above, we wanted the coefficients for aircraft ages near each other to be similar. This is because we have a prior belief that aircraft of similar ages pose a similar risk of accident (all other factors being the same). An early approach to this problem in the literature is called the "fused LASSO." In their work on this approach, Tibshirani et al. (2005) had n features that were "close" to each other, much in the same way that aircraft ages are close to each other—that a priori they expected coefficients to be similar. And they proposed to minimize

$$\min_{\beta}\left(-l(\beta)+\lambda_1\sum_{i=1}^{n}|\beta_i|+\lambda_2\sum_{i=2}^{n}|\beta_i-\beta_{i-1}|\right).$$

The first penalty term is the familiar LASSO penalty term, and the second term adds an extra penalty for changes of coefficient from one feature to the next. This is a very direct and clear way of ensuring that the coefficients for features near each other are the same if possible.

In the fused LASSO, the penalization added by a coefficient depends not only on itself but also on the values of the coefficients next to it. This creates complications in trying to find the optimal coefficients. The approach we have taken here is not to change the penalty term but to change the features going into the model.

In the paper behind the aglm R package (Fujita et al. (2020), Kondo (2021)), the authors show a strong connection between the fused LASSO and using step functions. As an example, consider the case where we have four aircraft only, with ages 0, 1, 2, and 3. There is an intercept which picks up the base frequency for age 0 and which is β_0. The coefficients for ages 1, 2, and 3 are β_1, β_2, and β_3. For convenience, we assume that $\beta_2 > \beta_1$ and $\beta_3 > \beta_2$. Under the fused LASSO, the total penalty is

$$\lambda_1 \times (\beta_1+\beta_2+\beta_3)+\lambda_2\times[\beta_1+(\beta_2-\beta_1)+(\beta_3-\beta_2)].$$

(The first term inside the λ_2 multiplication is the full value of the first coefficient since there is no previous coefficient.) If we set λ_1 to zero, the penalty is $\lambda \times \beta_3$.

Using step functions, we would have steps at 1, 2, and 3. If we choose the intercept to be the same and the parameters to be β_1, $\beta_2-\beta_1$, and $\beta_3-\beta_2$, then we will get exactly the same predictions, so the likelihood will be the same. In addition, the penalty will be $\lambda\sum|\beta_i| = \lambda\times\beta_3$, which is also the same.

Hence we see that our use of step functions is the same as the fused LASSO with $\lambda_1 = 0$.

5.9. Next Steps

From the results in Section 5.6 we now have a reasonable model for the FAA-NTSB data, with nonlinear effects being fitted and with a better performance than our baseline model. However, it excludes all features that are high cardinality categorical variables. In the next section, we will consider how we might help our GLM to model such features.

6. High Cardinality Categorical Variables (HCCVs)

6.1. Introduction

Generalized linear models (GLMs) require every feature to be in numeric form. This is because there is always a stage whether in training the model or in creating predictions where each feature is multiplied by a numeric coefficient. However, in the GLM that we built in the last section, there are 15 categorical variables—such as the type of aircraft, type of engine, and type of registrant. For example, two of the engine types are

- Turboprop[26] (code "2")
- Rotary[27] (code "11")

and our fitted model does have coefficients for these: −0.09 for turboprop engines and 3.58 for rotary engines. Since the GLM could not take the codes or descriptions of engine type as features, the software we are using first created additional numeric columns to deal with these features, and it was these features that it provided to the GLM fitting process. For example, it created a column called "faa_eng_type.11," which is 1 whenever the engine type is rotary and zero otherwise. It did the same for all other engine codes and also for all levels of any other factors. The extra columns are sometimes called dummy variables, and this process is known as one-hot encoding. The 15 categorical features in this model had between them 99 levels. To represent these levels, the GLM model matrix used 84 columns of 0/1 indicators. One level from each categorical feature does not require a separate column as it is captured by the intercept.[28] In the full FAA-NTSB data there are 2 million rows, and the 84 new columns of the model matrix require 900MB of memory. Actually, since the new columns consist only of zeros and ones, we can use a sparse way of storing the matrix, which only stores the ones and the extra storage required reduces

[26] Turboprop engines are fitted on fixed-wing craft and use a gas turbine to drive a propeller.

[27] Light engines with a high power-to-weight ratio. In our data, these are often used in home-built or custom-built aircraft.

[28] When fitting non-penalized GLMs, this approach is required. When using penalized regression, all 99 levels can be used.

to 90MB. In our data there are 97 different countries, 16,304 different cities, and over 40,000 different makes of aircraft. These are referred to as HCCVs because they have a large number of levels.[29] If we were to use the same one-hot encoding approach for these variables, we would need a large amount of memory. Given that this is not a particularly large dataset and HCCVs could be far bigger cardinality than this, the memory issues are potentially significant.

In this chapter, we discuss methods for handling HCCVs. The central idea is to shift from representing each HCCV level with a separate column (resulting in hundreds or even thousands of columns) to encoding each level with one or a small number of values (leading to just one or at least only a small number of columns).

6.2. Target Encoding

If the one-hot approach to HCCVs would create models that performed well on future data and there was absolutely no other way to deal with the issue, then we would certainly find a way to overcome the large memory requirements seen above. For example, we can train models on one row of the data at the time (this is known as online learning). We, therefore, further consider the impact of using one-hot encoding for HCCVs.

Adding a huge number of one-hot features to our data and then using unpenalized regression is likely in general to encourage overfitting and to lead to unreliable parameter estimates for levels of the feature which have few observations. If we were to use penalized regression, then these problems could possibly be overcome. Consider the simple case of Gaussian regression with an identity link function and using only one feature, which is an HCCV. A small amount of algebra (see Section 6.8.1) suggests that, given the one-hot approach, it is not unreasonable to use the following coefficient for the k'th one-hot column (which is the one-hot representation of the k'th level of the HCCV):

$$\beta_k = \frac{(\bar{y}_k - \bar{y})}{\frac{\lambda}{n_k} + 1} \quad \text{for } k = 1, 2, \ldots, p, \tag{6.1}$$

where β_k is the coefficient, n_k is the number of ones in the one-hot column (which is the same as the number of observations in level k), there are p levels (where p is large), $\bar{y}$ is the overall average response, $\bar{y}_k$ is the average response for the kth group, and λ is the

[29] There is no particular dividing line for when to use one-hot encoding and when to use the methods discussed in this chapter. In fact, some software have a parameter for the number of levels below which one-hot encoding is used and at or above what another method will be used. The optimal level for this parameter can be chosen by cross-validation.

regularization parameter chosen by cross-validation. The predictions would then be

$$\bar{y} + \frac{(\bar{y}_k - \bar{y})}{\frac{\lambda}{n_k} + 1},$$

which can be rearranged as

$$\frac{\frac{\lambda}{n_k}}{(\frac{\lambda}{n_k} + 1)}\bar{y} + \frac{1}{(\frac{\lambda}{n_k} + 1)}\bar{y}_k.$$

The outcome of this approach seems reasonable. For example, assuming that λ is fairly high, if we have only one observation for a given level of the factor, the coefficient for this group will be shrunk to close to zero, and the prediction will be the estimate of the overall mean. However, if we have a very large number of observations in a group, the coefficient for that group will be close to the difference in the mean level of the response for that group and the overall mean.

The formula for predictions, being a weighted average of the overall mean and the mean for each level, with the weight on the mean for a level increasing with the number of observations, is familiar to us as a credibility estimate. We show in Section 6.8.2 that the penalization parameter plays a similar role to the ratio of the "between group variance" to the "within group variance." (An early version of this estimator is known as the James-Stein estimator and is discussed in the very interesting note by Efron and Morris (1977).) There is, however, a clear difference in the approach. In the credibility theory, the ratio of variances[30] is assumed to be the optimal parameter. Here, we do not claim to know, a priori, what is optimal. We find λ which optimizes average cross-validation performance. So long as our validation data is representative of future data and is large enough to give stable results, we may end up with better generalization performance than the traditional credibility approach.

While the manner in which we derived the above simple result does not apply where there is more than just the one HCCV feature in the penalized regression or for cases besides the Gaussian identity link, the general idea of some amount of shrinkage between the overall mean and the group mean, depending on the number of observations in each level of the factor, will be preserved.

Based on the above, besides the huge memory problems, it seems like penalized regression would deal with HCCVs well. However, there is another issue. The λ which we need to be using to deal with the HCCV might not be the same level of penalization

[30] In Empirical Bayes Credibility/Nonparametric Credibility one estimates the expected value of the process variance and the variance of hypothetical means from the data.

required to optimally deal with all the other features which are not HCCVs. This could possibly be dealt with using a versions of penalization discussed in Section 4.6.6, such as the adaptive LASSO or the grouped LASSO. Overall, it seems that using the one-hot approach with penalized regression would lead to reasonable results, but it might present some significant technical or practical problems along the way.

Given Equation 6.1, it does seem that we can achieve results very similar to one-hot penalized regression on HCCVs in a different way. For each level of the categorical feature, x_k, $k = 1, \ldots, p$, we replace x_k by

$$x'_k = \frac{(\bar{y}_k - \bar{y})}{\frac{\lambda}{n_k} + 1} \quad \text{for } k = 1, 2, \ldots, p$$

and use this feature in the penalized regression.[31] This simple approach resolves all the technical challenges mentioned above. In our FAA-NTSB case, this would be very simple, for each make of aircraft we calculate the difference between average frequency for that make and the overall average frequency, shrink it towards zero, and then use those values as a new feature in the model.

This type of encoding is often called "Target Encoding" or "Mean Encoding." Care needs to be taken when using software with such methods to make sure that there is some credibility adjustment, since implementations will often only replace the HCCV with the mean response (e.g., the mean accident frequency in our case). We note also that where there is a natural hierarchy, the complement of credibility can be chosen to reflect it. This is discussed this further in Section 6.4.3.

The discussion above was based on an additive model. In the absence of significant credibility for level k, we shrink the value of the target encoded feature for level k (x'_k) to zero. We can be more explicit about the shrinkage towards zero by adding a term where zero is the complement of credibility and it is multiplied by 1 minus the credibility factor. Equation 6.1 then becomes

$$x'_k = \left(\frac{\frac{\lambda}{n_k}}{\frac{\lambda}{n_k} + 1} \right) \times 0 + \left(\frac{1}{\frac{\lambda}{n_k} + 1} \right) \times (\bar{y}_k - \bar{y}) \quad \text{for } k = 1, 2, \ldots \tag{6.2}$$

In a multiplicative world, in the absence of significant credibility, we shrink the value

[31] Note the difference in the letter used in this formula compared to the almost identical formula 6.1. Previously the formula gave a suggestion as to what might be a reasonable coefficient to be fitted to a one-hot column representing the k'th level of our HCCV. That is why 6.1 uses $\beta_k = \ldots$. Here, we are creating a new feature, hence the use of the letter x'. With appropriate choices for λ, it may turn out that the value for the k'th level of the new feature is similar to the coefficient that it would have been fitted as the k'th one-hot column in a GLM.

of the target encoded feature for level k to one. This suggests that we replace the equation above with

$$x'_k = \left(\frac{\frac{\lambda}{n_k}}{\frac{\lambda}{n_k} + 1} \right) \times 1 + \left(\frac{1}{\frac{\lambda}{n_k} + 1} \right) \times (\bar{y}_k / \bar{y}) \quad \text{for } k = 1, 2, \ldots \tag{6.3}$$

If our target, y, is the number of claims, then $(\bar{y}_k/\bar{y})$ is the ratio of average claims frequency in group k to the overall average claims frequency. $(\bar{y}_k/\bar{y})$ is an example of the ratio of actual to expected experience since $\bar{y}_k$ is the actual experience for the group and $\bar{y}$ is the expected experience for the group (under a simple intercept-only model). We will refer to this as the actual versus expected (AvE) ratio. If level k has a high AvE ratio and a large number of observations, it will be represented by a high value of the target encoded feature.

An alternative formulation to 6.3 is to shrink the log of the AvE ratio towards zero

$$\log(x'_k) = \left(\frac{\frac{\lambda}{n_k}}{\frac{\lambda}{n_k} + 1} \right) \times 0 + \left(\frac{1}{\frac{\lambda}{n_k} + 1} \right) \times \log(\bar{y}_k / \bar{y}) \quad \text{for } k = 1, 2, \ldots$$

In the case study in Section 6.6, we will be using the log of the target encoded feature in the GLM. This formulation gives that log directly and is potentially better suited to its use in this context.

In any case, when using the log of the target encoded feature with the log link function, then the multiplicative contribution of this feature to the prediction is $x_{te}^{\beta_{te}}$, where β_{te} is the coefficient fitted to the target encoded feature.[32]

There remains the practical issue of how to choose λ. This is discussed in Section 6.6.

6.3. Leakage

We give our GLM various useful features like aircraft age, make, and model, and we essentially ask the GLM to find a function which takes these inputs and to provide as an output the estimate of the claims frequency. This function is not necessarily going to be very accurate, but we expect that on future data it will perform better than a naive or baseline model. In order to help the GLM get accurate results during training, we could just give it a new feature which, for each observation, is the value of the target it is trying

[32] See Goldburd et al. (2025) Figure 1 and Section 2.4.1 which states: "When a log link is used, it is often appropriate to take the natural logs of continuous predictors before including them in the model, rather than placing them in the model in their original forms." In our case, if β_{te} ends up being close to 1, then the final relativity for each level of the target encoded feature is close to its (shrunk) AvE. This seems reasonable and is an outcome which is easy to communicate to underwriters.

to predict. In this case the GLM would get perfect results during training by ignoring all other features and just setting the prediction to be equal to the target variable which we have given it. This model would be totally useless—it could not actually make a prediction on future observations since for future observations we don't yet know the value of the target. And any useful relationships between the other features and the target will not be picked up because the GLM can get a zero training error just using the one target feature that we gave it.

Allowing the GLM to see the target or part of the target during training is called leakage. And given that it is guaranteed to make our model less good, we try to avoid it.

In the above approach (target encoding), we have introduced leakage. We allow the GLM to see the average value of the target for each level of the HCCV. While this is not quite as bad as allowing the GLM to see the value of the target for each observation, it is not far off—since for HCCVs many of the factors may only have one observation or a small number of observations. The solution to this is that for observations in any one of our validation folds, we only allow the target encoding to use observations from all the other folds.

For example, consider the aircraft manufacturer (known as "make" in our dataset). This is certainly an HCCV, having over 40,000 levels in our dataset. If for one of the makes, we have six aircraft split into three folds as in Table 6.1, then the target encoded feature for fold 1 is the average frequency for folds 2 and 3, which is $\frac{3}{4} = 0.75$. Likewise the feature value for fold 2 is $\frac{2}{4} = 0.50$ and for fold 3 it is $\frac{1}{4} = 0.25$.

Table 6.1. Target encoding for each fold.

		across other folds		
fold	accidents	aircraft	accidents	frequency
1	0	4	3	0.75
1	0	4	3	0.75
2	0	4	2	0.50
2	1	4	2	0.50
3	1	4	1	0.25
3	1	4	1	0.25

We end up therefore with a separate target encoding for each fold. In the above example, the target encoding for fold 1 is estimated only using data from the other two partitions. We now create a model using the data from folds 2 and 3 and the target encoding for fold 1. We can now safely validate performance on fold 1 data, since the target encoding used did not see fold 1 data.

By the time we are finished training our model, we will have one column on the training data which is a target encoding of the HCCV. However, its value for a given make

will depend on the fold the observation is in. When we wish to predict on test data, we need a single value for the target encoding for each make. We therefore create one final target encoding on all the training data, which we can then use when doing any final tests of our model.

Table 6.2 shows the raw accident frequency (before credibility adjustments) for some aircraft makes based on all training folds and also the raw frequencies we would use for folds 1 to 3 (e.g., the frequencies shown for fold 1 are calculated excluding the aircraft in fold 1). We can see that the accident frequency for the aircraft with more exposure in our dataset is fairly stable but for those with less exposure there is some variance depending on which fold we leave out. However, we can see that even when there is some variance, there is also consistency—the accident frequency for Socata is consistently high and that for Learjet is consistently low.

Table 6.2. Raw accident frequency (before credibility adjustments) for some aircraft makes.

		train	train frequency excluding		
faa_acft_make	ex	frequency	fold_1	fold_2	fold_3
CESSNA	440577	0.0046	0.0045	0.0046	0.0046
PIPER	277902	0.0046	0.0046	0.0046	0.0046
BEECH	108464	0.0045	0.0045	0.0044	0.0045
BOEING	45257	0.0040	0.0040	0.0042	0.0039
MOONEY	30744	0.0033	0.0033	0.0034	0.0034
WACO	4181	0.0022	0.0025	0.0022	0.0020
AIRBUS INDUSTRIE	4164	0.0041	0.0048	0.0034	0.0036
SOCATA	4075	0.0066	0.0060	0.0069	0.0069
EMBRAER	4069	0.0020	0.0020	0.0023	0.0014
ENGINEERING & RESEARCH	4051	0.0012	0.0014	0.0014	0.0009
LEARJET INC	3877	0.0015	0.0012	0.0018	0.0015

6.4. Some Other Encoding Approaches

6.4.1. Grouping rare categories

A much simpler approach than target encoding is to combine categories that appear infrequently into a single "Other" category. This approach will reduce the number of categories so that we can safely revert to one-hot encoding and penalized regression. It also avoids us having to explicitly choose the credibility weighting part of the target encoding. If most of our data is in a small number of categories and the rest of our data very sparsely populates a large number of other categories, this approach will group together levels of

the factor for which our credibility adjustment would probably have been zero anyway.

In grouping rare categories, the practitioner may be able to use their own domain knowledge to determine which levels "belong" together, e.g., should all rare makes from the same country or continent be grouped together and kept separate from rare makes from other countries or continents?

The idea of grouping rare levels as suggested here may also be used to improve the results from target encoding. When target encoding is used on a rare level, the credibility adjustment will lead it to be encoded at the value of the complement of credibility. If domain knowledge is used to group similar rare levels into a larger category, the resulting group may be a reasonable size and the target encoding will be able to reflect the experience of the group.[33]

6.4.2. Frequency encoding

In various machine learning and modeling software that deal with HCCVs, an approach called frequency encoding is included. It does not address our key issue, which is how does each individual level of the HCCV relate to our target variable? We do include it here, though, for completeness and in case it is encountered during modeling.

In frequency encoding we replace each category with the proportion of the training data that belongs to that category. Table 6.3 shows this type of encoding on our data; the feature faa_acft_make (which a GLM cannot use directly) is turned into faa_acft_make_fe, a numeric feature, which the GLM can use.

Table 6.3. Frequency encoding of aircraft make. 25.88% of the aircraft are Cessna, 16.21% are Piper, and so on. The GLM can directly use the numeric feature faa_acft_make_fe.

faa_acft_make	faa_acft_make_fe
CESSNA	0.2588
PIPER	0.1621
PILATUS AIRCRAFT LTD	0.0020
DASSAULT AVIATION	0.0019
SMITH	0.0009

While frequency encoding does not solve our original problem, it might provide a useful feature. For example, the appearance of a make of aircraft very infrequently in our dataset may imply something about the riskiness of the aircraft or of the type of people who choose to fly such an aircraft.

[33] In workers compensation, for example, very small classes are often grouped with other classes for ratemaking purposes and are thus charged the same rate.

6.4.3. Hierarchical grouping

Micci-Barreca (2001) discuss cases where the HCCVs have some hierarchy. They give various examples:

- Zip codes sharing the same first three digits are typically located within the same metropolitan area.
- Standard Industry Codes (SIC), which are used to classify commercial organizations based on their primary line of business. For example, code "4512 – Air Transportation, Scheduled" is part of major group "45 – Transportation By Air," which itself is part of group "4 (known as division E) – Transportation, Communications, Electric, Gas, And Sanitary Services."
- The first three digits of a telephone number often represent a natural grouping, such as an area or a specific service (such as a toll-free number).

Even where a formal hierarchy does not exist, the practitioner may be able to use domain-specific knowledge to create one.

Once we have a hierarchy, we could simply replace the HCCV with the broader categories and then use them as one-hot features in a GLM. However, we can do better than that. We can continue with the target encoding approach above but use as a complement of credibility not the overall mean, but the mean of the value encoded for a higher level of the HCCV, which has more data. Micci-Barreca (2001) discuss approaches to hierarchical HCCVs in more detail.

If we are going to use a hierarchy, especially as a complement of credibility, we need to be exceedingly careful that it is meaningful in the correct way. For example, consider manufacturer (make) and model of motorcycle when creating rates for motorcycle insurance. This is certainly a hierarchy—each model belongs to the category of its manufacturer. However, in terms of helping to predict the target variable using this hierarchy, it would be a very bad idea. Yamaha manufactures the YZF-R1 and YZF-R6, which are high-performance sport bikes. It also manufactures mopeds and scooters used for commuting, such as Yamaha Vino and Yamaha Jog. In a dataset used for rating, the average performance for Yamaha will be the average across their full range (to the extent it is in the data). Imagine now that only a small number of the high-end sport bikes are insured. Their credibility factor will be low, and if we use manufacturer as the complement of credibility, their final rate will reflect the average Yamaha performance and will be much too low.[34]

[34] In a few years, after significant insurer losses (due to adverse selection), the majority of Yamaha bikes in the portfolio will be high-end sports bikes and the price will finally be corrected.

6.4.4. Feature creation and word embeddings

In this section, we give an outline of two methods that can be used when the practitioner is faced with an HCCV. If an approach along these lines seems promising for a given line of business and HCCV, one could first apply it and then use target encoding in a second stage. The idea of splitting modeling into two stages is discussed in Chapter 8.

Target encoding is a supervised learning approach—it explicitly looks at the target variable in our data. The approaches outlined here create additional features, manually or automatically, without directly looking at our data, and then use these columns in a GLM.

As an example, consider insuring businesses against the risk of fire, where SIC is one of our features. We can ask risk engineers and other domain experts to explain the key drivers of fire claims and their cost. We might find that they can be split between the risk of inception of a fire and the risk of spread of a fire. We might further find that inception risk is greatly increased for businesses which use application of heat in their processes (main street restaurants as opposed to a main street clothing stores). Likewise, the risk of spreading a fire can be significantly increased for businesses that store combustible materials or those that create dust as part of some process. Further discussions might lead to 10 questions which seem to be most important in driving the frequency or severity of fire claims. These questions can then be asked of domain experts for each SIC code.[35] We now include in the GLM 10 new features—the responses to the 10 questions,but not the HCCV itself. It is highly likely that these will be predictive.

Another non-supervised approach is based on "word embeddings." These are the backbone of any neural network which involves language (for example, large language models). Each word is represented as a vector. The vectors are derived so that words used in similar contexts have vectors which are close together in some way and are known as word embeddings. Today, the most advanced large language models make their underlying word embeddings available, and it is therefore relatively simple to get the word embeddings for every make or model of aircraft. The embeddings we use here each consist of 1,536 numbers (dimensions), and in Table 6.4, we show the first three numbers for three different aircraft.

[35] In the past, these conversations were time intensive and required the collaboration of a wide range of experts from around the business. Today, with appropriate prompts, large language models can sometimes create draft output relatively quickly.

Table 6.4. The first three numbers (dimensions) in word embeddings for the three aircraft listed.

make	model	position_1	position_2	position_3
PIPER	PA-28 Cherokee	−0.021193	−0.007403	−0.006449
PIPER	CUB	−0.021803	−0.012660	0.014906
CESSNA	Skyhawk	−0.013765	0.000489	−0.003003

A measure called "cosine similarity" can be used to check whether two vectors are similar. For this measure, an outcome close to one means the vectors are similar (point in the same direction), and an outcome close to zero means that they are very different. In our case, the output is 0.131 when comparing the Piper Cherokee to the Piper Cub and 0.108 when comparing the Cherokee to the Cessna Skyhawk. This does not seem particularly useful as the Piper Cherokee is actually more similar to the Cessna Skyhawk (in landing gear and usage) than it is to the Piper Cub.[36]

Given the (long) word embeddings, we can apply principal component analysis to represent their key features in a smaller number of dimensions. Finally, we can use these features in a GLM. There is no guarantee that these features—having been derived in an unsupervised manner—will help our GLM to pick up the key aspects of the underlying HCCV. However, this approach and the previous more manual approach have proven useful to the authors in some cases.

6.5. Generalized Linear Mixed Models

Generalized linear mixed models (GLMMs) are discussed in Goldburd et al. (2025), Section 10.1, and in West et al. (2022).[37] In this section, we first motivate the need for GLMMs with examples from various lines of business. We then briefly revisit how they are formulated and finally consider how we can use them as a solution to modeling HCCVs.

The fitting of GLMs relies on an independence assumption: the value of the target variable (y_i) for one observation must not influence or depend on the value of the target variable for any other observation (e.g., y_j) in the dataset, given the model's parameters (coefficients). This assumption is crucial because it allows the model to be applied to each observation individually and it underpins the use of techniques like maximum likelihood estimation. Consider the following lines of business:

[36] The generation of word embeddings is complex and understanding what their 1,000 or more dimensions represent is hard. The overall process described here is therefore somewhat opaque. Nevertheless if the resulting features are useful, we can ask an underwriter to sense-check them and provide adjustments as needed.

[37] At the time of writing, *Linear Mixed Models: A Practical Guide to Using Statistical Software* (West et al. (2022)) is required reading for MAS II.

- An insurance product which covers high-net-worth individuals who own and drive many cars. Our dataset consists of one line per car. Each line of data contains significant information about the type of car and also a unique identifier for the individual. If a high-net-worth individual owns 10 cars, then 10 observations will have that person's unique identifier.
- Fleet insurance: A product which provides auto cover for a pool of cars which companies provide to their employees. The cars in each company's pool are varied, and our data contains one line of information for each car within each company but limited or no information is available about the drivers. A unique company identifier is associated with each company. If one company has 100 cars in its fleet and was insured for three years, then 300 observations will have that company's unique identifier.
- The same as Fleet insurance above, but now there is a unique identifier for each company and also for the division within the company.

In all of the above cases, can we use a linear model or a GLM to find the coefficients for the features associated with the car? Yes. Clearly, we can. There is nothing to stop us. The question is: should we, though?

Imagine that we have fitted a model to all the features about the cars. Consider a company for which we have 100 observations. Were we to look at 50 of those observations and find that the residuals are high, does that tell us anything about the other 50 observations? Yes, it does. It suggests that the employees of that company are higher risks than average and therefore that the other 50 observations will also have high residuals. This means that, given the coefficients, the target variables are not independent between observations. The independence assumption that we mentioned above is not met.

To overcome the above problem, we could add the company identifier (or the company and division identifiers) into the GLM. This would of course be an HCCV because we insure many companies and the identifier takes a different value for each company. This leads us immediately to another problem. If we want to insure a new company that we have not insured before, our historic data will not have the unique identifier for that company, and so we cannot rate new companies. We can get around this, too, but all of these problems suggest that a different approach is warranted.

We can model the features for car in the normal way but treat the impact of the company on the level of risk as a random effect which varies from company to company. This random effect will increase or decrease the intercept term for each company but will not affect the coefficients for the vehicle features. If we build our model in such a way that this random effect has a mean of zero, then for future companies, in the absence of information and experience about the company, it will be reasonable to assume that the random effect is zero.

In the above model, the coefficients for the vehicles are not considered random. They are the same across every observation and are known as fixed effects. Overall, then we have a mix of fixed effects and random effects in the same model. Such models are known as linear mixed-effects models.

In these types of models—known as (generalized[38]) linear mixed models—the resulting behavior of the coefficient for the HCCV is similar to target encoding, i.e., it is pulled towards the central tendency of the response, in proportion to the quantity of the data in the level of the HCCV. However, while the effect might resemble target encoding, the underlying mechanics and statistical reasoning are different. Additionally, there are practical tools and software available that implement these models and thus they can be used as a useful check on target encoding results.

Given p fixed effects (e.g., p features that we know about each car) and an intercept term β_0, we add the random effects (a_k) to our model specification as follows:

$$
\begin{aligned}
y_{ik} &= \beta_0 + \beta_1 x_{1k} + \beta_2 x_{2k} + \cdots + \beta_p x_{pk} + a_k + \epsilon_{ik} \\
a_k &\sim \mathcal{N}(0, \sigma^2_{company}) \\
\epsilon_{ik} &\sim \mathcal{N}(0, \sigma^2),
\end{aligned}
$$

where a_k is the random effect for company k and is distributed with a normal[39] distribution mean 0 and variance $\sigma^2_{company}$.

Historically in random-effects models, statisticians were not necessarily interested in the value of the fitted random effects. They just wanted to make sure that they were dealing with the structure of the dependency between observations correctly. In the above example, we would certainly be interested in the values of the fitted random effects, as it would inform the rate for renewals of existing clients. Fortunately, open-source software is available which fits these models and provides estimates of both fixed and random effects.

Given the above, we can treat aircraft make as a random effect, and we will then get estimates for the impact of aircraft make on frequency from the fitted random effects. We could also treat the aircraft model as a random effect within each aircraft make.[40]

---

[38] The difference between linear mixed models (LMMs) and generalized linear mixed models (GLMMs) is analogous to the difference between linear models (LMs) and generalized linear models (GLMs).

[39] A reason for using the normal distribution for random effects is that we have no a priori belief on how the experience for individual companies will be different from average performance, but we expect some to be worse than average and some better in a symmetric way.

[40] As per our previous discussion in the section on hierarchical grouping, this is not necessarily a sensible thing to do.

6.6. Case Study—HCCVs

As part of our FAA-NTSB case study, we applied both target encoding and GLMMs to our HCCVs to see if this would improve performance. We had hoped for a large performance improvement, but cross-validation performance was only slightly improved. Real-life data does not always do what we expect it to.[41] The authors have found that, in practice, dealing properly with HCCVs does normally improve performance, and we therefore proceed to demonstrate these approaches on our case study, using aircraft make and model as an example. (For our target encoding approach, we used overall means and not make as the complement of credibility.) The main effort in target encoding is to find an appropriate level for λ, and we now discuss this in some detail.

As mentioned above and as we will show below in the technical section, the λ parameter in the target encoding section can be thought of as the ratio of the "within group variance" to the "between group variance." (In Bühlmann credibility, the terms used are "variance of hypothetical means" and "process variance"; and the ratio is referred to as Bühlmann's k.)

"Within group variance" is the variance of the random number of accidents (for one aircraft over one year) for aircraft within a given make and model. We can get a rough idea of the magnitude of this by reasoning that the number of accidents for one aircraft is a 0-1 Bernoulli random variable[42] and that the typical mean accident rate is the overall mean accident rate in our training dataset, which is 0.004584. The mean of the binomial distribution with probability p is p and the variance is $p \times (1 - p)$, therefore the variance is $0.004584 \times (1 - 0.004584) = 0.004562987$.

"Between group variance" is the variance of the true differences in mean accident frequency between aircraft makes and model. For example, if we only had three aircraft (equally common in our data) and we knew that, compared to the average frequency, they have accident frequencies of 0.1% less, the same, and 0.1% more, then the between group variance would be the variance of the three numbers −0.1%, 0, and +0.1%. If we knew the true group means we could easily calculate this variance—but then we would not be doing all this work to estimate them (i.e., our target encoded HCCV will essentially be our best estimate of the group means). However, given that we already have a GLM which generates predictions excluding make, we can find the mean residual by make. If

[41] A while back, a colleague told one of the authors that the predictions from an analysis should be used because the method was statistically sound. Cross-validation is a harsh taskmaster and gives us an additional framework within which to carefully consider whether our models will improve performance. On the other hand, we should not be slave to cross-validation performance—this is discussed further in the final chapter.

[42] Equally, if we are concerned about the very small number of craft which actually had more than one accident in a year, we could stick with our Poisson assumption and assume a variance equal to the mean. The difference is small.

we had a huge number of observations for each make and model, these mean residuals would be a reasonable estimate of the effect of make on accident frequency. We don't have a huge number of observations for each, but we can take only those makes for which we have a reasonable number of observations. Doing this gives an estimate of the between group variance of 7.038492×10^{-6}.

Finally, the ratio of within group variance to between group variance is

$$0.004562987/(7.038492 \times 10^{-6}) = 648.$$

This gives an idea of the magnitude of λ. This would mean, for example, that if we have 1,000 aircraft-years of experience for a make, then we would give $\frac{1000}{1000+648} = 0.61$ credibility to claims experiences for that make.

If we simply used $\lambda = 648$, we would now calculate the target encoded feature for each fold and then run our GLM including all our previous features and the new target encoded feature. In general, though, we prefer to use our cross-validation procedure to choose the best features and best models, so instead, we carry out a model fitting process for a range of values of λ (4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384),[43] and for each value we calculate the average validation performance. The result is shown in Figure 6.1. It can be seen that performance improves (slightly) compared to the model without HCCVs. The dashed vertical line shows the performance at $\lambda = 648$. We can see that this is, in fact, close to the best average cross-validation performance, but the best improvement is for $\lambda = 1024$ (or 2048).

The actual feature we used in our GLM was the log of the target encoding discussed above. Given the log link we have

$$log(\text{ predicted frequency}) = \text{intercept} + \cdots + (\beta_{te} \times \log x_{te}),$$

where x_{te} is the target-encoded feature and β_{te} is the fitted coefficient.

Taking the exponent of both sides, we have

$$\text{predicted frequency} = k \times \cdots \times x_{te}^{\beta_{te}},$$

where k is the exponent of the fitted intercept.

---

[43] The values were chosen simply to span a significant range, with our estimate of 648 somewhere in the middle. If a graph of the results would have shown the need for more details in one area of this range, we would have rerun the analysis accordingly.

[43] The results of using credibility are relatively insensitive to misestimates of K. See "An Actuarial Note on Credibility Parameters" by Mahler (1986).

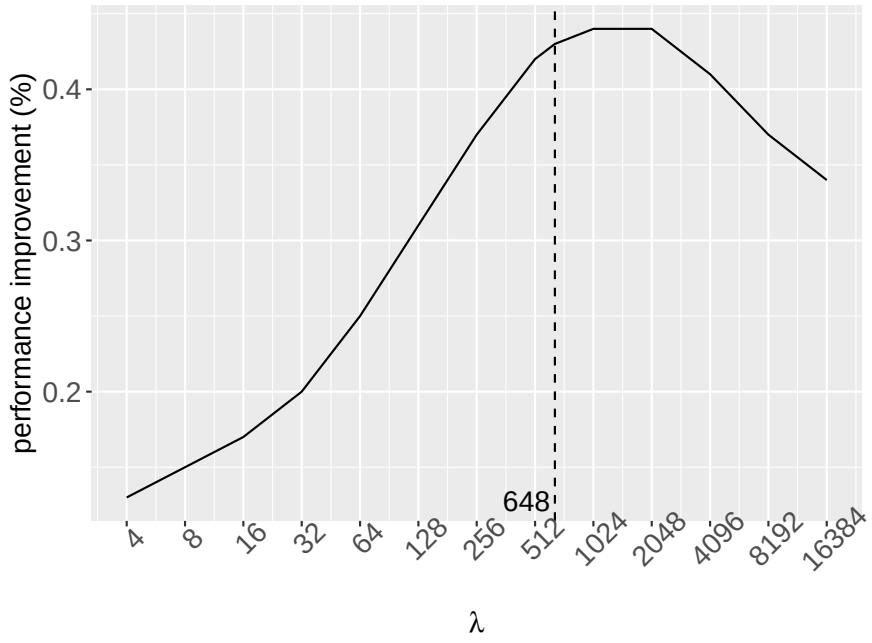

Figure 6.1. The percentage increase in performance when including the target encoded aircraft make and model feature using different values for λ. The dashed vertical line shows the performance at $\lambda = 648$. This is close to the best average cross-validation performance but $\lambda = 1024, 2048$ are slightly better (see footnote).

In practice we often find that, if we have carried out the target encoding process correctly and there are no data issues, then the value of β_{te} associated with optimal cross-validation performance is not far from 1. We also expect that as λ increases, β_{te} increases. We therefore inspected the β coefficient in the GLM for the target encoded feature as a function of λ.

Figure 6.2 shows the coefficients for each value of λ. As we expected, we see that β_{te} increases as λ increases. Optimal cross-validation performance is around $\lambda = 1024$ or $\lambda = 2048$, and the associated value of β_{te} are 0.4 and 0.5. Increasing λ to 4096 would produce a coefficient closer to 1 but would lose performance as cross-validation performance is not flat near the maximum.[44] Therefore we proceeded with $\lambda = 2048$. The target encoded feature was included in the GLM, and the average cross-validation performance increased from pseudo-R^2 0.150 to 0.151.

[44] Usually, given that the λ associated with the best cross-validation performance is somewhat lower than 1, we would inspect our data to see if any levels of the HCCV have large amounts of exposure relative to the others and are dominating the fitting process. In our case, the actual changes in performance (rather than the percentages) are all rather small and so we have not investigated further.

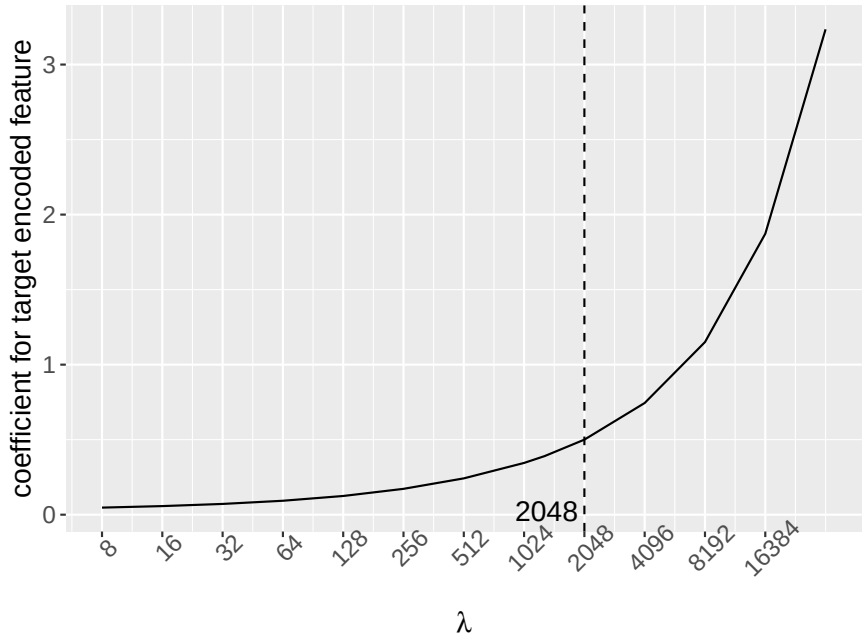

Figure 6.2. The y-axis shows the fitted coefficient for the target encoded feature. A value close to 1 would suggest that we used a reasonable level of credibility. The coefficient at $\lambda = 2048$ is not close to to 1.

Section 6.5 introduced GLMMs. Open-source software is available which fits such models and provides estimates of both fixed effects and random effects. An advantage to the GLMM approach is that it extends naturally to dealing with hierarchical variations in risk, such as aircraft model within aircraft make. We therefore expected to be able to easily use this approach and that performance would be similar to the target encoding approach used above. In practice, we were not able to achieve consistent results using this approach. In particular, the software we were using gave error messages when fitting. Given that target encoding has the added advantage that it stays within our overall cross-validation framework and that its performance did exceed any of the GLMM models we were able to fit, we proceeded with target encoding.[45]

[45] We would, nevertheless, encourage the interested reader to try GLMMs in this and similar situations as they can be a very useful tool. They do also have the advantages being part of a statistically sound framework and of taking only a very small number of lines to code given the packages readily available for them.

6.7. Practically Speaking

6.7.1. Sense-checking the results and ad-hoc adjustments

Within an insurance company, pricing would be done by or in conjunction with people who understand the risks being dealt with. We can use the results of our encoding to highlight areas that need further research. In early work not presented above, we used target encoding at the level of make only. Table 6.5 shows the two highest and two lowest values for the target encoding. For this output, we would want to understand more about DJI: why do they have zero claims given the large amount of exposure. We research DJI and find out that they manufacture drones. We can then talk to subject matter experts to understand why they have zero accidents entered in the NTSB database and whether or not it makes sense to include their data in our analysis.

The two highest adjustments in Table 6.5 are cases which have one claim but only a small amount of exposure and, therefore, a very high AvE ratio. Taking "STUMP GREAT LAKES," for example, the AvE ratio is 223.75. With $z = \frac{12}{12+1024} = 0.011583$ the credibility calculation then gives the target encoding as $1\times(1-0.011583)+222.22\times 0.011583 = 3.58$.

Table 6.5. Examples of the output of target encoding aircraft make. ex and num_cl are exposure and the number of claims, respectively. AvE ratio is the ratio of the GLM predicted frequency to the actual frequency. z is the credibility factor based on the value of λ that we chose. make_te is the target encoded value. In 11,900 aircraft-years of exposure for the DJI make, there were no accidents. This compares to 3.692 accidents expected from the model. The credibility adjustment of 0.92 leads to a target encoding of 0.079, which then gets used in the GLM. Makes with one claim and very low exposure have a very high AvE ratio value, so that even though z is low, the value of the target encoded variable is very high.

aircraft make	ex	num_cl	AvE ratio	z	make_te
DJI	11900	0	0	0.9208	0.079
TEXTRON AVIATION INC	5483	12	0.24356	0.8426	0.363
MARLIN JOHN E	13	1	186.59373	0.0122	3.266
STUMP GREAT LAKES	12	1	223.74936	0.0116	3.579

Now, imagine discussing with an underwriter why you have proposed to more than triple the rate for this make on the basis of seeing exactly one claim. You will try arguing that on average your approach is a good approach, but it will not be an easy conversation. While our approach works on average, there are some edge cases where we or others do

not feel comfortable in implementing the proposed outcomes.[46]

The approach we have taken is to make an ad hoc adjustment to the credibility given to the group mean where the expected number of claims is small (e.g., 0.005) and the number of claims is also small (e.g., 1).[47]

The edge cases can be driven simply by the randomness of claims experience (as in our case above), in which case a slight adjustment to the general approach seems reasonable. However, edge cases can also be driven by data quality or operational issues. For example, a system may by default give a driver who is not listed on a policy but who legally drives the car one day of exposure—simply because the true amount of exposure is not known and a default is needed. If this driver has a claim, the annualized frequency will be high. Across all such records, the frequency will be very high. The practitioner will need to deal with such situations on a case-by-case basis.

6.7.2. Target encoding using GLM predictions

We have discussed target encoding using as the actual versus expected ratio, $(\bar{y}_k/\bar{y})$, the ratio of the mean response for group k to the mean overall response. Given that we already have a GLM excluding the HCCVs, there is an alternative. We can calculate the predictions $(\hat{y}_i)$ for each observation within a particular make and model and then calculate their average $(\bar{\hat{y}}_k)$. Finally, we proceed as above but using as the actual versus expected ratio, $(\bar{y}_k/\bar{\hat{y}}_k)$. As an example, say that the average overall frequency $(\bar{y})$ is four accidents per thousand aircraft-years, but Bell helicopters (helicopters are included in the data) have an accident frequency of 16 accidents per thousand years of experience, then the first approach above would set the target encoding at 4[48] —which is very high. On the other hand, our GLM includes a feature that differentiates between fixed-wing and rotor-wing craft. This feature will have already picked up that helicopters generally have a high claims experience. It might therefore be that the average prediction for Bell helicopters is 20 per thousand. The target encoding on the second basis mentioned above would be 0.80.[49]

Intuitively, it feels right to first allow less complicated features, such as aircraft age and number of engines to pick up the key patterns, and then to allow the target encoded HCCV to pick up any remaining inaccuracies in the model predictions. However, it is certainly more work as we first need to fit a GLM, then create the (k-fold) predictions, and then fit another GLM.

---

[46] This is a general situation that can occur when applying credibility methods.

[47] The manner of this adjustment is not particularly important—we chose to reduce the exposure used in the credibility calculation so that it remained at zero until a certain threshold had been passed. The main point here is that a method that performs well on average can leave dangerous weak points in certain areas of the rating plan, and great care is needed.

[48] $\bar{y}_k = 16$ and $\bar{y} = 4$.

[49] $\bar{y}_k = 16$ and $\bar{\hat{y}}_k = 20$.

Our approach in the case study was indeed to create the target encoding using the GLM predictions. Indeed any eagle-eyed underwriters reading this paper might have noticed that the higher end of the target encoded variable did not include helicopters. Had we been using the overall mean as the expectation in our AvE ratio, helicopters would be likely to feature (since the actual frequency of helicopter accidents is much higher than the average accident frequency) (see Table 6.6).

Table 6.6. Target encoding using the overall mean as the expectation in the AvE ratio. Helicopters tend to have higher frequencies than fixed-wing craft and indeed appear at the higher end.

faa_acft_make	ex	num_cl	AvE ratio	z	make_te
DJI	11900	0	67.5796	0.9197	0.0803
AERONCA	27451	45	136.2547	0.9640	0.3544
ROBINSON HELICOPTER CO	5693	122	30.4230	0.8475	3.5511
ROBINSON HELICOPTER	7513	153	39.1231	0.8800	3.5615

6.7.3. Novel categories in the future

What happens if a new make of aircraft, which is not in our historic data, appears in the future and requires us to provide a rate? There will be no value for the target encoded feature for this make. One choice is to set the value of the target encoded feature to the average of value of this feature across all makes and to produce a quote. This would seem to be a particularly bad idea. If make is an important part of our rating, we are completely guessing that the right price for this make is the average price across all makes. There is no reason at all for this to be true. If the make is associated with high risk, we will leave ourselves open to massive adverse selection. It would seem far better to set up our systems not to offer a quote and to flag to the administrators that a new make of aircraft has been encountered. Discussions can then be had and deliberate value for the feature can be chosen based on similarities of the new make with existing makes.

6.7.4. What does zero mean?

What does zero mean for a target encoded feature? (Or a one in a multiplicative structure.) It can mean two very different things.

- That we have lots of exposure and, across this, the mean experience for the make is very close to the mean experience for all makes. We are very sure that the true experience of this make reflects the average experience.
- We have very little experience, and regardless of how experience for this make looks compared to average, we are ignoring it because we don't know if it is a fluke or not.

We are completely unsure as to whether the true experience of this make reflects average experience.

This is truly a problem. If we blindly use rates from target encoded HCCVs we run the risk of pretending that we understand all the types of risk when we are actually totally uncertain about some of them. A systematic and statistical approach to this would be to set our prices not at the mean predictions from our GLM but at some other (higher) percentile. More simply though, given that there are (hopefully) people in the insurance business who understand the risk, the derived rates and relativities should be discussed with them. This is particularly easy if there are existing rates for the make, as any significant changes (for makes with small amounts of exposure) can then be discussed.

6.7.5. Keeping up with other ideas

In order to keep up with ideas of how to approach HCCVs, it is useful to look at what is implemented in popular platforms. For example, Python's scikit-learn package has a section called category encoders (sklearn_category_encoders) which implements (as of the date of writing) 20 approaches to encoding categorical variables. Simply reading the documentation for each of them gives a good overview of the types of approaches currently being used.

6.7.6. Read the documentation

Applying methods and functions written by third parties into open-source software like R and Python is easy. It is not always easy to understand what the software is doing. Reading the documentation helps. Sometimes documentation is not perfect, though, and some further digging is required. For example, at the time of writing this monograph, an alternative software has a method to create target encoding. An example of how it is called (from R) is:

```
target_encoder <- targetencoder(training_frame = df_,
x = 'faa_acft_make',
y = "resid",
fold_column = "fold",
data_leakage_handling = "KFold",
blending = TRUE,
inflection_point = 500,
smoothing = 100)
```

We can quickly get a rough idea of what most of these arguments mean, but "inflection_point" and "smoothing" are not clear. The documentation tells us that the inflection_point "determines half of the minimal sample size for which we completely trust the estimate based on the sample at the particular level of the categorical variable" and

that "smoothing controls the rate of transition between the particular level's posterior probability and the prior probability. For smoothing values approaching infinity, it becomes a hard threshold between the posterior and the prior probability." This is more clear, but we have to dig into the code of the alternative software to find that the formula actually implemented is

$$z = 1.0/(1 + \exp((\text{inflection\_point} - n_k)/\text{smoothing})).$$

In Figure 6.3 we look at two examples of this function, compared to the credibility estimate we have been using. The solid black line is our usual formula $\frac{n_k}{n_k+\lambda}$ where $\lambda = 500$. The dashed line shows the formula implemented with inflection_point = 500 and smoothing = 100. We can see that the size at which 50% credibility is given is the same, but the behavior elsewhere is quite different from our standard formula. If we increase the smoothing parameter to try to make the behavior more similar in the middle, we find that about 25% credibility is given where size = 0. Given that we would always hope that credibility tends to zero for group sizes close to zero, we should not automatically expect this formula to perform well. Indeed, various tests with different values of the arguments for this formula did not provide good average validation performance.

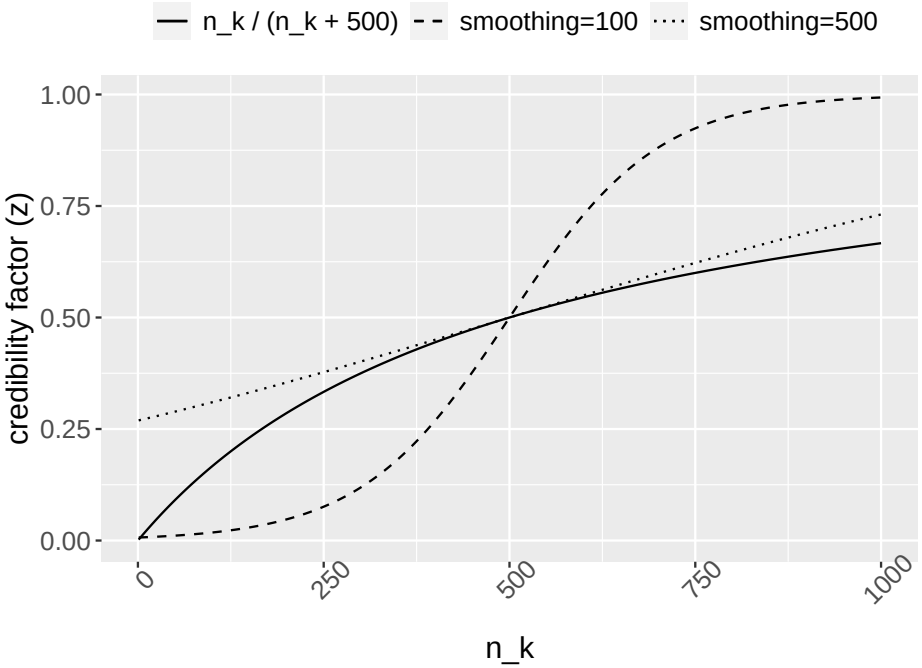

Figure 6.3. A comparison of the credibility function we have been using (solid back line) to a function implemented in one software for target encoding. We can see that the shapes are quite different.

Overall, we see that it is always worth checking which formula are being implemented by the methods and functions we use.

6.7.7. Are HCCVs really predictive?

The risk of accident arises from the pilot (human error), the aircraft (failure of the machine), and the environment (e.g., the runway surface, weather, etc.). Many of the features that one might use are clearly linked to one (or more) of these sources of accident. For example, pilot age, flight hours on the same type of craft, total flight hours—all clearly link to the risk of pilot error. Aircraft make and model clearly link to the aircraft. However, if we also have in our model full details about the maintenance of the aircraft, its age, and the number of type of engines, we might think that there is no longer any need to include the make and model itself since every possible feature that directly describes the risk of failure of the machine is already in the model.

We nevertheless will always include the make and model (and other HCCVs) in our GLM. Essentially, we "let the data do the talking." Regardless of what we might think, provided that we do actually want the best performing model, if average validation performance across all folds improves, we will be better off with aircraft make and model in the GLM. If performance does improve considerably it is worth trying to understand why. In particular, if we see that one level of an HCCV with significant volume is very predictive, we should check if it is due to one particular feature and then find a way to populate that feature across the whole dataset. Alternatively, that level of the HCCV may represent a class of risks which are so different from the rest of the data, they are best priced in a different model.

HCCVs can also be important because they pick up some residual information about the machine (or whatever they are describing) or because they are proxies for some other source of risk. A careful (or risky) person may tend to pilot certain makes and models of aircraft.

6.7.8. Sense-checking the data

We have been talking about there being over 40,000 (43,617 in fact) different makes of aircraft in our data. There are not 43,617 different manufacturers of aircraft. A quick look at the name of the make of craft for which only one has ever been built shows the names to be those of individual people. The field in our data "aircraft builder certification code" takes the value of "1: Not Type Certificated" for almost all these craft. These are amateur-built aircraft. The number of different makes including only the "0: Type Certificated" category reduces to 994, which seems more reasonable. Even within these data, there are some craft with only one record, however, the make seems to indicate a commercial manufacturer (e.g., AIRBUS HELICOPTER, BOEING/HUMM STAN-

LEY, CESSNA-LOOMIS-BUEHN, etc.).

Once again, therefore, our data preparation has let us down,[50] this time in two ways. Firstly we should have realized upfront that aircraft make includes many individuals rather than aircraft manufacturing companies. And even within the commercial manufacturers we might want to relabel some to group them together. For example, given that we saw AIRBUS HELICOPTER above, we then looked at the different makes with the name AIRBUS in them and we found: "AIRBUS," "AIRBUS CANADA LP," "AIRBUS CANADA LTD PTNRSP," "AIRBUS DEFENCE AND SPACE LTD," "AIRBUS DEFENSE AND SPACE S A," "AIRBUS HELICOPTER," "AIRBUS HELICOPTERS," "AIRBUS HELICOPTERS DEUTSCHLAND," "AIRBUS HELICOPTERS INC," "AIRBUS INDUSTRIE," "AIRBUS S A S," and "AIRBUS SAS." We might actually be happy for each of these "makes" to have different prices arising from different values of the target encoding—but we should certainly allow that only after discussing the data with a person who understands the different parts of AIRBUS.

6.8. Technical Note

6.8.1. HCCV estimates using Ridge regression

In this section, we derive the Ridge regression estimates of coefficients in a Gaussian regression with identity link where we have only one HCCV and no intercept.

Assuming our HCCV has p levels, we have p columns from the one-hot encoding, with each column having its own coefficient, $\beta_1, \ldots, \beta_p$.

We can imagine sorting our data so that all the observations with level 1 of the factor come first. There are n_1 of them with values $y_{11}, y_{21}, \ldots, y_{n_1 1}$.

In general, for group k we have coefficient β_k and y values $y_{1k}, y_{2k}, \ldots, y_{n_k k}$.

The loss function is essentially made up of the sum of squared errors of each level plus the penalty. Previously we have put the fraction $\frac{1}{2n}$ before the sum of squares—this is typically done to simplify the algebra—it makes no difference to the solution path since $2n$ is a constant. We could equally remove the $\frac{1}{2n}$ fraction and multiply λ by $2n$ and get the same solution. In this case, we will use a fraction $\frac{1}{2}$ as the result will then look simpler.

[50] And we are reminded that an actuary needs to understand the data with which they are working. The actuary should understand the situation being modeled, with help from others when necessary.

$$
\begin{aligned}
l(\beta) =& \frac{1}{2}\sum_{i=1}^{n_1}(y_{i1}-\beta_1)^2 \\
&+\frac{1}{2}\sum_{i=1}^{n_2}(y_{i2}-\beta_2)^2 \\
&\cdots \\
&+\frac{1}{2}\sum_{i=1}^{n_p}(y_{ip}-\beta_p)^2 \\
&+\frac{1}{2}\lambda\beta_1^2+\frac{1}{2}\lambda\beta_2^2+\cdots+\frac{1}{2}\lambda\beta_p^2
\end{aligned}
$$

Only one of the βs appears in any one term, so when we differentiate with respect to any one β all the other terms disappear. For example, if we differentiate with respect to β_1 we get

$$\frac{\delta l(\beta)}{\delta \beta_1} = -\sum_{i=1}^{n_1}(y_{i1}-\beta_1)+\lambda\beta_1.$$

Setting this to zero at the minimum gives:

$$\sum_{i=1}^{n_1} y_{i1} = (n_1+\lambda)\beta_1$$

and

$$\hat{\beta}_1 = \frac{n_1\bar{y}_1}{(n_1+\lambda)}$$

i.e.,

$$\hat{\beta}_1 = \frac{\bar{y}_1}{(1+\frac{\lambda}{n_1})}.$$

The same is true for any other coefficient, and so we have

$$\hat{\beta}_k = \frac{\bar{y}_k}{(1+\frac{\lambda}{n_k})}.$$

As discussed in the main text, having no intercept is not a good idea. For some large λ, as we get fewer observations at one factor level, we would prefer to predict the overall mean and not zero. Hence, it seems reasonable to carry out the above penalized regression after first deducting the mean response from every value of y. This way, the coefficients

tell us how far the prediction for a level should differ from the overall mean, and we have

$$\hat{\beta}_k = \frac{(\bar{y}_k - \bar{y})}{(1 + \frac{\lambda}{n_k})},$$

and our prediction for group k is

$$\bar{y} + \frac{(\bar{y}_k - \bar{y})}{(1 + \frac{\lambda}{n_k})}.$$

This can be rearranged to give

$$\frac{\frac{\lambda}{n_k}}{(1 + \frac{\lambda}{n_k})}\bar{y} + \frac{1}{(1 + \frac{\lambda}{n_k})}\bar{y}_k.$$

As an aside, we note that the above is not what would result from including an unpenalized intercept term and carrying out penalized regression on the y values without deducting the mean. In that case the penalized regression would ensure that the one-hot coefficients summed to zero—and that is not the case for our solution above, where there are different numbers of observations for each factor (which is pretty much always).

6.8.2. Link to the Bayesian approach

In our approach to model fitting so far, we have thought of each individual coefficient (each separate β) as a fixed value which we are trying to estimate. The penalization ensures shrinkage and/or feature selection. Although we do not discuss it in any detail here, it is known that carrying out a Bayesian regression with an appropriate prior on each coefficient separately and then choosing the maximum a posteriori (MAP) estimate will lead to the same estimates as Ridge regression or the LASSO (see for example the original LASSO paper, Tibshirani (1996), Section 5).

There is a similar Bayesian approach which is particularly useful for HCCVs. We start off with a prior belief that the true different levels of risk for each make of aircraft are independent and are distributed according to a normal distribution with mean zero and variance of τ^2. We assume that the coefficients for each level of the HCCV are independent. Our prior is therefore

$$\beta_k \sim \mathcal{N}(0, \tau^2) \quad \text{for } k = 1, 2, \ldots, p.$$

Consider a model where the only feature is the HCCV. Within each group, we have

n_k observations $y_{1k}, y_{2k}, \ldots, y_{n_k k}$ which are distributed as follows:

$$\begin{aligned} y_{ik} &= \beta_k + \epsilon_{ik} \\ \epsilon_{ik} &\sim \mathcal{N}(0, \sigma^2) \\ \beta_k &\sim \mathcal{N}(0, \tau^2). \end{aligned}$$

Using Bayes' theorem,

$$Pr[\beta|y, X] = Pr[y|\beta, X]\frac{Pr[\beta|X]}{Pr[y|X]}.$$

Given that $Pr[y|X]$ is some fixed constant and that the prior distribution of β does not depend on X, we have

$$Pr[\beta|y, X] \propto Pr[y|\beta, X]Pr[\beta].$$

Taking the log of each side and noting that $Pr[y|\beta, X]$ is the likelihood of the observed ys given β and X (which we will refer to as $L(\beta)$) we have

$$\log Pr[\beta|y, X] \propto \log L(\beta) + \log Pr[\beta],$$

where $Pr[\beta|y, X]$ is the posterior distribution of the βs given the data.

Firstly, let us look at $\log L(\beta)$. Within any group (k), given the value of β_k, the probability of the value y_{ik} is

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_{ik} - \beta_k)^2}{2\sigma^2}\right).$$

Ignoring constants and taking logs, for one observation, y_{ik},

$$\log L(\beta) \propto -\frac{(y_{ik} - \beta_k)^2}{2\sigma^2}.$$

Summing over all observations in each group and over all groups, we have

$$\log L(\beta) \propto -\frac{1}{2\sigma^2} \sum_{k=1}^{p} \sum_{i=1}^{n_k} (y_{ik} - \beta_k)^2.$$

As for $\log Pr[\beta]$, this is just the product of the p independent prior distributions for the βs and is $\propto -\frac{1}{2\tau^2} \sum_{k=1}^{K} \beta_k^2$.

Therefore

$$\log Pr[\beta|y,X] \propto -\frac{1}{2\sigma^2}\sum_{k=1}^{p}\sum_{i=1}^{n_k}(y_{ik}-\beta_k)^2 - \frac{1}{2\tau^2}\sum_{k=1}^{K}\beta_k^2.$$

To find the value of β_k which maximizes this posterior, we differentiate and set equal to zero:

$$\begin{aligned}
&\frac{d}{d\beta_k}\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n_k}(y_{ik}-\beta_k)^2 - \frac{1}{2\tau^2}\beta_k^2\right] = 0 \\
\implies &\frac{1}{\sigma^2}\sum_{i=1}^{n_k}(y_{ik}-\beta_k) - \frac{1}{\tau^2}\beta_k = 0 \\
\implies &\sum_{i=1}^{n_k}(y_{ik}-\beta_k) - \frac{\sigma^2}{\tau^2}\beta_k = 0 \\
\implies &\sum_{i=1}^{n_k} y_{ik} - n_k\beta_k - \frac{\sigma^2}{\tau^2}\beta_k = 0 \\
\implies &\beta_k\left(n_k + \frac{\sigma^2}{\tau^2}\right) = \sum_{i=1}^{n_k} y_{ik} \\
\implies &\beta_k = \frac{\sum_{i=1}^{n_k} y_{ik}}{n_k + \frac{\sigma^2}{\tau^2}}.
\end{aligned}$$

If we express the average value of the target variable within each group as

$$\bar{y}_k = \frac{1}{n_k}\sum_{i=1}^{n_k} y_{ik}$$

we can then write

$$\beta_k = \frac{n_k\bar{y}_k}{n_k + \frac{\sigma^2}{\tau^2}}$$

and divide the top and bottom by n_k to get

$$\beta_k = \frac{\bar{y}_k}{1 + \frac{\sigma^2}{n_k\tau^2}}.$$

6.9. Next Steps

We have now trained a GLM on our FAA-NTSB data. We avoided overfitting, dealt with numeric features which affect the target in a nonlinear way, and dealt with HCCVs. Yet our performance (pseudo-R^2 of 0.151) still falls 9% short of our gradient boosting benchmark (pseudo-R^2 of 0.165). In Chapter 9 we discuss how we can use the gradient boosting model (or any complex machine learning model) to check the work we have done so far with our GLM and how we can try to extract any patterns it has found which we have not used and incorporate them into our GLM.

There is a choice with how to deal with our target-encoded HCCVs which we did not discuss above. The first option is to refit the GLM including the HCCV together with all other features previously used. If we do this, we will get fitted coefficients for the target encoding variable and also revised fitted coefficients for all the other features. They may be different from the previous model, and indeed, unless the other features are completely independent of the target encoded features, it is likely that they will be different. The other option is to allow the coefficients from the first model to remain as they were and to fit the target encoded HCCVs in a separate, second-stage model.

In a business which has a complex GLM which drives its prices, it may not want to have to implement changes in every table of coefficients just because it carries out a detailed review of one of its features.

The choice we made for our modeling in this chapter was to fit our target encoded variable in a second-stage GLM and so to avoid changing the coefficients for all the other features. The method for achieving this has many applications, and we discuss it in some detail in Chapter 8. However, we first take a break from the FAA-NTSB study and turn to the issue of modeling in the presence of highly correlated features.

7. Highly Correlated Variables

The issue of highly correlated variables in data is discussed in Goldburd et al. (2025) (Section 2.9, Correlation Among Predictors, Multicollinearity and Aliasing). In this chapter, we discuss this topic in more detail and illustrate some of the points with a worked example. We will begin by identifying the scenarios that can lead to our data having high correlations. Then we will try to understand why this matters at all—why not just fit a model and proceed as normal? Finally, we will discuss the various approaches to dealing with the issue.

So far in this monograph we have focused on fitting GLMs. The issue we address in this chapter is more general, and therefore before we proceed, it is worth stepping back a little and considering the general process of how to approach supervised learning tasks.

Supervised learning refers to any situation when our data contains a history of many examples with their outcomes. Typically, each example is a row in our dataset and has features (e.g., policyholder age, aircraft make and model) and the outcome (e.g., accident or no accident). We want our model to find a relationship between the features and outcomes on the historic data such that when we apply that relationship to future data, it accurately predicts outcomes when we don't yet know the outcome. Some of the stages in any supervised learning task are:

- Which type of model should we use? This could be a linear model, a tree-based model, or a neural network.
- Which model should we use? If we had decided to use a tree-based model, we could then choose from decision trees, random forests, gradient boosting machines, and their various derivatives. If we had decided to use a linear model, we might decide to use a GLM with penalization and the LASSO. This and the previous choice may simply be performance driven. That is, we will train many model types and choose the one with the best validation performance.
- Given that we already have some data with features, should we try to augment the features and if so how? This is a surprisingly hard task. It includes trying to create or purchase data assets relevant to the problem and trying to engineer new features from features that we already have. This process is called "feature engineering." It is

not limited to purchasing and engineering features when training the model, we also have to ensure that identically defined features are available when the model is applied operationally.

- Finally, given the model type we are going to use and the set of features in the training data, we may wish to select only certain features to use in training our model or to appear in our final model. This is called "feature selection."

Issues involving highly correlated variables tend to relate to one or more of these stages, and we will refer to them in our discussion below. Much of what we cover will be related to feature engineering and selection. For a much more detailed coverage we refer the reader to the excellent and easy-to-read book, *Feature Engineering and Selection, A Practical Approach for Predictive Models* (Kuhn and Johnson (2019)).

7.1. How Do Highly Correlated Variables Arise?

7.1.1. Naturally

Correlation between features in our data can arise naturally. Let us consider a portfolio of young drivers in auto insurance and a jurisdiction where the minimum age to take a driving test is 17. When preparing rates for auto insurance, two features that will often be in the data are driver age and driver experience. Age is often recorded as the driver's age rounded down to the nearest whole number at the policy start date. Driving experience is often recorded as the number of years since passing the driving test, rounded down to the nearest integer, at the policy start date. (For younger drivers or those with more limited driving experience, rounding down to the nearest month might result in better performing models). If every driver passes their test shortly after their 17th birthday and immediately buys a car and takes out insurance, then every 17-year-old in our data will have zero years' experience since passing their test, every 18-year-old will have one year, and so on. Driver age and driving experience will be perfectly correlated. In practice we should certainly expect to see among younger drivers a high correlation between driving experience and age.

As an aside, we note that whether or not two features are correlated will depend on the type of insurance and how they are measured. For example in General Aviation insurance, if we are measuring pilot experience as the number of flight hours in the type of aircraft they are insuring, then we would not necessarily expect this to be highly correlated with age—a pilot might move from one type of aircraft to another at various points in their career.

Part of getting to know the data one is working on is to check for highly correlated features. Where we see high correlations when we were not expecting any (and vice versa) we have an opportunity to increase our understanding of the data by finding out why

this is the case.

7.1.2. As a result of data enrichment

There are very many situations where data enrichment may lead to highly correlated features. Here are some examples:

- In locations where regulation permits use of credit score in rating plans, data might be enriched not only with credit score but also with all the underlying features that are used to create the score. It would be surprising not to find high correlations both between these features themselves (e.g., monthly income, home ownership, average account balance, etc.) and between them and the credit score.
- More generally, enrichment can consist of thousands of new features, many of which measure similar things. A feature may related to whether or not the insured has had certain types of court judgments against them and if so, how many. This feature might be present a few times, being measured (say) over the past three months, six months, one year, and three years. These features will be highly correlated.
- Enrichment features for geographical analysis may include the distance to the nearest school, the nearest police station, and the nearest fire hydrant. These may all be correlated with population density and with each other—hence, we may find some of these features to be highly correlated.

As mentioned above, before using data, correlations should be checked and the reasons for high correlations should be understood. This is especially true for data enrichment sources which might be coming from third parties—understanding the reliability of such data is particularly important.

7.1.3. As a result of feature engineering

To "engineer features" means to create new columns of data (features) which are based on the original columns that were provided. We have met feature engineering a few times in this monograph. When we took a single numeric feature and added many step or hinge functions, we were engineering features to allow our GLM to express a particular relationship. This is known as one-to-many feature engineering. We also discussed adding the distance between two points (e.g., address of the policyholder and the address where their vehicle is garaged) as a new feature. Later in this chapter we will replace all the numeric features in the example we will be working on with a different set of features (that are not correlated with each other). That is known as many-to-many feature engineering.

It is up to the modeler together with expertise from the business to ensure that any potentially useful features that the model would not otherwise engineer for itself are created and added to the data. Sometimes this process may lead to having highly correlated

features. Consider the case of commercial aviation insurance. Each insured is a commercial airline operating a fleet of craft (e.g., United Airlines, Southwest Airlines). For each fleet our data might contain one line per aircraft per year, with details of the aircraft and of any accidents or incidents it was involved in, together with the cost. As of 2024, Southwest Airlines has about 800 aircraft in its fleet. For each year that we have data on the fleet we will have about 800 lines of Southwest Airlines information in our data. We will certainly want to ensure that we add to each line information about the company itself and its attitude towards safety and risk management, pilot training, and so on. One aspect which is of interest and which might tell us something about the fleet and its operations is the range or spread of certain aircraft features across all aircraft. An airline which has an old fleet will have all old aircraft. As it starts to renew the fleet, the range of ages will increase. If the fleet is completely renewed, all the aircraft will be relatively new. Hence we may wish to add features related to the range and spread of aircraft ages. We would probably add range, interquartile range, standard deviation, and then the same measures (range, interquartile range, etc.) but split by the different types of aircraft on the fleet. We might also apply the same approach used for aircraft age to the value of the aircraft (incorporating features such as range, interquartile range, and standard deviation). At the end of this process we will have created many features, and given that they measure very similar things, we will expect them to be correlated.

7.2. Why Do They Matter?

There are various reasons why having highly correlated features in our data (or our final model) might be problematic:

- computational issues
- performance
- interpretability and communication
- cost

7.2.1. Computational issues

All supervised models are trained by trying to minimize a loss function (or sometimes to maximize a likelihood). If the loss function is curved near a minimum, it will be easy to identify the model parameters which achieve this minimum. However, if two features are highly correlated, their effects in the model can be traded off against each other—small increases in one parameter can be offset by small decreases in the other, leading to very similar predictions. This makes the surface of the loss function nearly flat in certain directions near the minimum. As a result the minimum will be hard to find.

The extent to which the above issue is a problem depends on the model type being trained.

The procedures used to fit GLMs can be badly affected by this issue. A non-penalized GLM might simply return some kind of warning or error and not return any fitted model. If it does return a fitted model, the parameters may be unstable. For example, two highly correlated parameters on which the target does not depend should both return a coefficient of zero, but it is possible that one would return −50 and the other 50. In addition, the confidence intervals around the parameters will be large. Penalized regression should always return an answer, but it might take a very long time.

Tree-based models should not suffer from large offsetting coefficients. The reason for the difference between GLMs and tree-based models in this matter lies in the way they are trained. Procedures to fit GLMs often consider how best to change all the variables at once by small amounts in order to get closer to the minimum of the loss function. Where two features are correlated, they can get changed in opposite directions at the same time with no effect on the loss function.

Tree-based models, however, are fitted in a "greedy" manner. This means that only the contribution of one feature is changed at the time. The tree has to make a decision on which feature to introduce a split on next. If two highly correlated features do not separately have an impact on the loss function then it is exceedingly unlikely to choose to introduce a split on either of them. And if they both have an impact, then it will choose one or the other—not both.

7.2.2. Performance

Kuhn and Johnson (2019) (Chapter 10, Section 10.3) consider the effect of irrelevant features on performance. In their example, the performance of penalized regression (LASSO) is not affected by irrelevant features, but all other model types that they look at (tree-based models, neural networks, and others) are affected to a greater or lesser extent. Our case, however, is different. We don't know whether the highly correlated features are irrelevant or not. We do know, however, that after including one of many highly correlated features, then the rest will be irrelevant. In the authors' experience, performance of tree-based model types is certainly adversely affected, and we will see below whether for a particular dataset GLM performance is affected—though a priori we would not expect it to be, since if the training of a GLM converges at all, it should converge to the minimum of the loss function.

In the presence of highly correlated features, even if the performance of the trained model is good, it relies on the same high correlation continuing into the future. If the correlation between the features changes, then the accuracy of the model may be compromised, even if no other element of the "ground truth" has changed. As an example, consider a case where a credit score and its five (say) components are all included as features when training a model. Credit score is a weighted average of the features with the

most significant weight on feature 1. The true relationship is between feature 1 and the target, but the trained model finds and uses the relationship between the credit score itself and the target. If in the future, the team that creates the credit score significantly reduces the weighting for feature 1, credit score will not be a proxy for feature 1 and model performance will suffer.

There is another, potentially much more dangerous way in which the correlation between features can change. This arises where customers can shop for different prices and the correlation has caused a "reversal"—that is to say that the coefficient is of the opposite sign than expected. For example, in private auto insurance (or in fact any insurance) we expect that, as the deductible reduces the price will increase. However, if the policy excess is very highly correlated with some other feature (for example, vehicle value), the fitted GLM might set the slope of the coefficient for vehicle value to be steeper than expected and the coefficient for policy excess to be upward sloping so that lower excesses have lower prices instead of higher prices. If the rating plan is available to the public or it is otherwise possible for customers to get different quotes for different excesses, customers with high value cars will very quickly start selecting low excesses, and losses will be made.

7.2.3. Interpretability and communication

If we are lucky, we might find that in the presence of many highly correlated features our GLM converges to the minimum of the loss function in a reasonable amount of time and the parameters look sensible. Nevertheless, we may still not want many highly correlated features in our model.

Consider a simple case where we are asked which features are important in our GLM. As long as the features are all on a similar scale, we can normally just look at the absolute coefficients in the fitted model. Features with large absolute coefficients are more important than those with low coefficients (as long as they both affect similar proportions of customers). In the presence of highly correlated features, however, we have to be much more careful. If we have 10 highly correlated features, all measuring driver experience in slightly different ways, then each of the coefficients may be small, but their overall effect on the predictions may be very large. If the effect of experience on the target variable is nonlinear, then the only way to find the overall shape is to somehow add up the effects of all of the correlated features. These difficulties are not insurmountable, but they add complexity in trying to understand our model and will certainly make it much harder to communicate to the business of feature importance and the shape of the effects.

In some cases, the standard approaches in the software we use will not work in the presence of highly correlated features, for example, the detection of interactions. One method of doing this is to see how often two features appear in consecutive decisions or

along one decision path in trees. Consider driving experience and the power-weight ratio of the car. We might expect these to interact (i.e., inexperienced drivers in a car with a high power-weight ratio are a much worse risk than the individual coefficients would imply). In this case, when a tree is split on power-weight ratio, we would often expect the next split (or another in the same path) to be driving experience. Across all the trees, we might find that 100 times, a path which splits on power-weight ratio also splits on experience. Yet, if 10 correlated features are used to measure experience, we may only get 10 paths for each of them, and this might not stand out compared to other combinations of features.

7.2.4. Cost

There is a cost to having a feature in our final model. It has to be collected at runtime either from the proposer or from enrichment sources. It has to be stored. Likewise, a table of coefficients needs to be created and stored. It has to be added to a calculation. In future training, it becomes part of data quality audits and exploratory data analysis. It is much easier to monitor model performance and data drift for 10 features than for 100 features. Some costs may "just" be time costs, others may be actual expenditure or the risk of making errors. Either way, it seems that our aim should be, in the words of Kuhn and Johnson (2019),

> *to reduce the number of features as far as possible without compromising predictive performance.*

7.3. Introducing a Case Study

We will illustrate some of the approaches we are about to discuss with the UK road traffic accident data mentioned in Section 2.2. To avoid issues with missing values, we use data for England only. In these data, England is split into about 34,000 areas (called Lower-layer Super Output Areas or LSOAs), each with around 1,000 households. In order to focus on local traffic, we exclude accidents on highways and similar types of roads. Figure 7.1 shows the number of accidents per quarter over the five-year period covered by the data. During 2018 and 2019 there were about 24,000 accidents per quarter. Data from 2020 and the first part of 2021 are heavily affected by the impact of COVID-19 on road traffic. The number of accidents per quarter after COVID-19 is about 10% lower than before COVID-19.

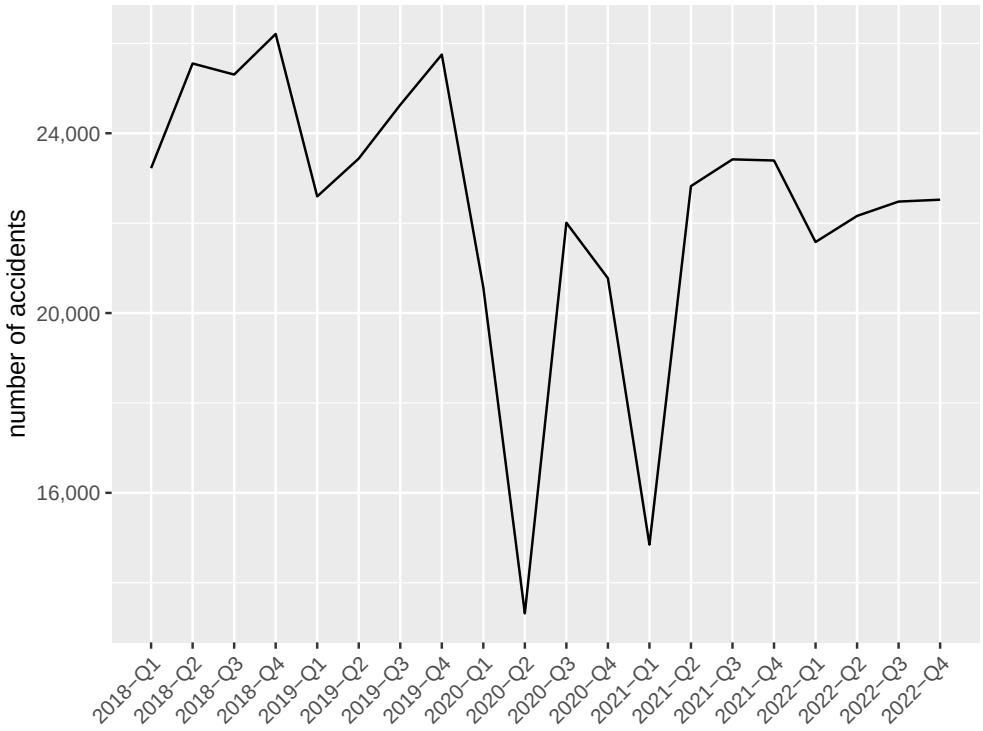

Figure 7.1. The number of accidents on the police database for England over the five-year period covered by the data. The data is affected by the impact of COVID-19 on road traffic.

We added various indices of deprivation and information from the 2019 UK census to the above data. These will be our features.

Examples of indices of deprivation are:

- crime_rank: Deprivation as measured by the risk of personal and material victimization. Low values indicate high levels of deprivation.
- income_rank: Deprivation as measured by the proportion of the population experiencing deprivation related to low income. The definition of low income used includes both those people who are out-of-work and those who have work but have low earnings. Low values indicate high levels of deprivation.
- employment_rank: Deprivation as measured by the proportion of the working-age population in an area involuntarily excluded from the labor market. This includes people who would like to work but are unable to do so due to unemployment, sickness or disability, or caregiving responsibilities. Low values indicate high levels of deprivation.

The Index of Multiple Deprivation (IMD) is also included as a feature. It is an overall relative measure of deprivation constructed by combining seven types of deprivation according to their respective weights. Unsurprisingly, it is highly correlated with some of

its constituent indices. For example, IMD and employment rank have a 0.91 (Pearson) correlation coefficient.

Census data measure various aspects for each household, such as:

- Legal partnership status
- Household composition
- Country of birth
- Age
- Distance traveled to work

Some of these aspects are measured both at a higher level and then broken down into detail. For example, the percentage of the population who are widowed or a surviving civil partnership partner is a feature. In addition, there are two further features: the percentage of the population who are widowed and the percentage of the population who are a surviving civil partnership partner. (This is very similar to how enrichment data will include many features that measure similar aspects of the risk.) This multiple measurement of very similar aspects of the population leads to highly correlated features.

In total, there are 70 features, of which 20 pairs (out of 2,415) have a 0.90 or greater (Pearson) correlation coefficient.

We define accident frequency as the number of accidents per 1,000 of census population. We should really define accident frequency as the number of accidents divided by the amount of traffic or the total distance traveled by cars on the roads. However, we have not added such information to our data. We therefore use population (of the LSOA) as a proxy.

We can see from the initial graphs that the accident frequency seems to vary with some of our features. For example, Figure 7.2 shows that accident frequency reduces for higher values (less deprivation) of crime rank.

Our aim is to build a model based on the available features and then to use it to predict the accident rate for areas not used in training the model. We will look at various methods of dealing with highly correlated variables and see if and how they impact our final model. As per usual, we will create a fold variable to be used for cross-validation during training and to exclude some data for testing. Given the time dimension of our data, there are one or two issues we need to think about when creating the fold variable, and these are discussed in Section 7.5.

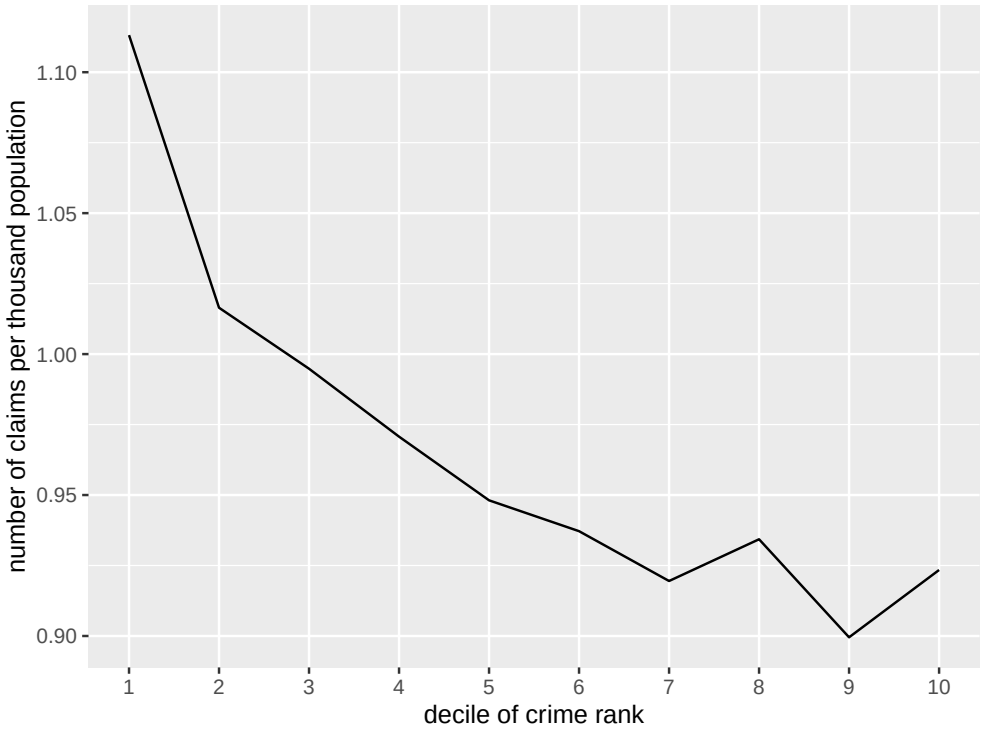

Figure 7.2. The relationship between crime aspect of deprivation (higher values means less deprived) and number of accidents per thousand of population.

7.4. Dealing with Highly Correlated Variables

7.4.1. Common sense

The original census data contains counts of the population in each category being measured. We prefer percentages because they lie in the range 0–100 and because they are easier to understand. We created percentages by dividing the number in each category by the total population. Similarly, the raw deprivation indices actually rank all of the LSOAs and therefore run from 1 to 34,000 (with the LSOA with value 1 being the most deprived). We grouped the LSOAs into 100 groups according to the percentile of their deprivation index.

After the above data manipulation, our census data contained both counts and percentages, and our deprivation data contained the original ranks and our derived groups. Every feature therefore had a pair with which it was perfectly correlated (this is known as intrinsic aliasing). Common sense tells us to exclude such features. We therefore removed the counts and the original ranks.

More generally, if an initial check of our data shows a pair of features with correlations close to one, it may be obvious that one should be removed.

Common sense has its limits though. Consider another issue with our data. For any aspect measured by the census, the sum of percentages across all categories must sum to 100%. Any one column can therefore be derived as a linear combination of the others in the same category (i.e., 100% minus the sum of all the other columns). This is an example of multicollinearity, and in non-penalized regression, it can lead to the same issues as having highly correlated features. We could exclude one of the columns, but it is not clear a priori which column to exclude. We therefore now consider other approaches. Note that for tree-based models, removing the column that represents one category is not a good idea. If the tree needs to split on that category, it will not be able to, and while it can split on all the other categories, this will lead to a more complicated model.

7.4.2. Removing some features before model training

A simple approach to dealing with highly correlated features is to remove some of them before training. One way to do this (as implemented in the R package, caret Kuhn (2008)) is to find the two most highly correlated features and, of those two, to remove the feature that has the highest mean absolute correlation with all other features. This is repeated until none of the remaining correlations is above a certain cutoff. Using this approach on our data (with a cutoff of 0.90) removes three of the deprivation indices (including the overall IMD) and nine census features.

If this approach is used, the value of the cutoff needs to be chosen. One approach would be to go train models for various values of the cutoff and for each to find the average cross-validation performance. Starting from a model containing all features, as we exclude more correlated features by reducing the cutoff we would expect cross-validation performance initially to either improve or at least not to deteriorate. The cutoff can be set at the level below which performance starts to deteriorate.

7.4.3. Ridge regression

The constraints created by penalization allow Ridge regression to find a set of parameters which maximizes performance, even in the presence of highly correlated features (or multicollinearity).

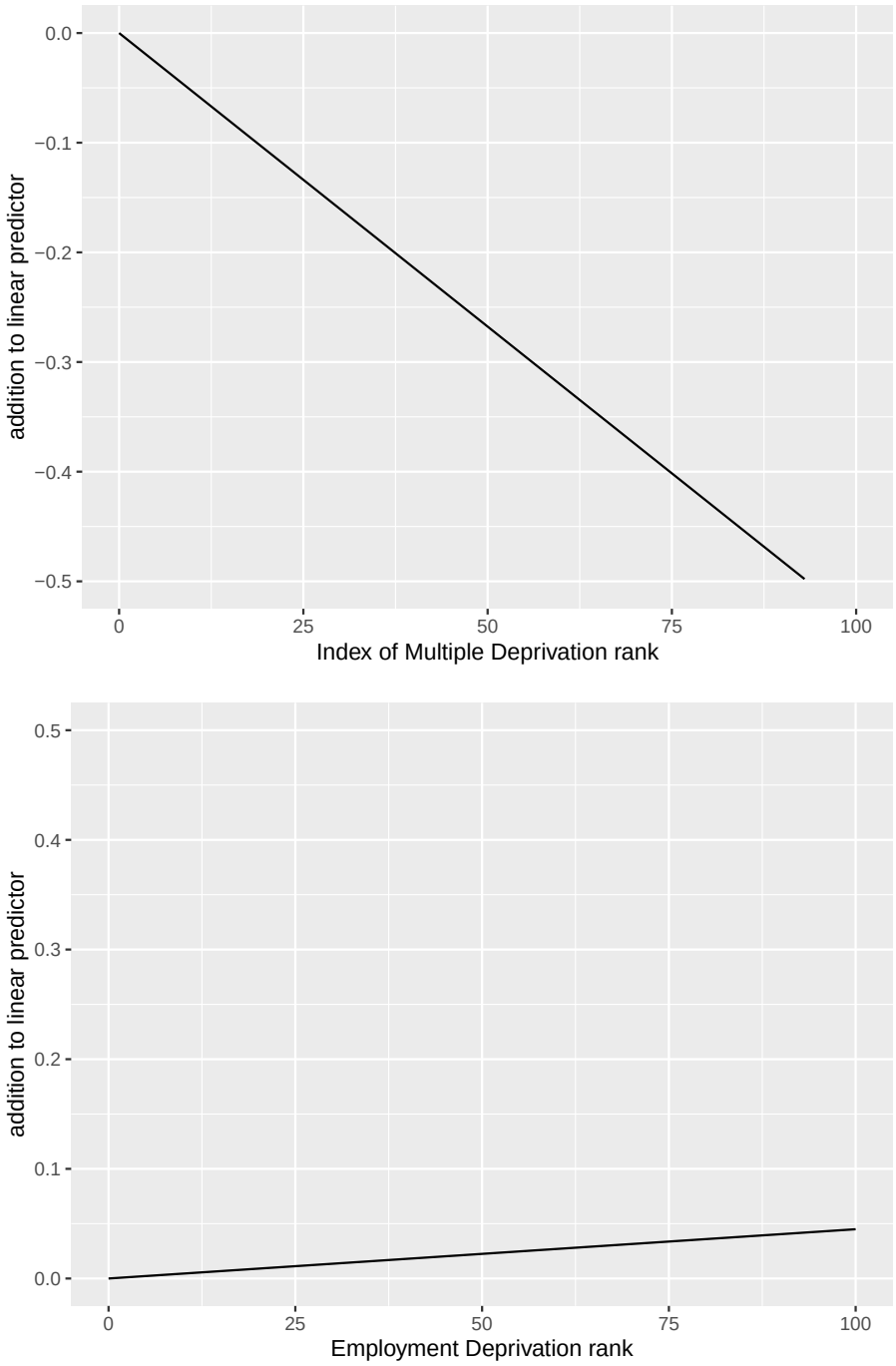

Figure 7.3. Coefficients from a Ridge regression model for IMD and Employment Deprivation Rank have different signs. This means that their contribution to the linear predictor cancels out to some extent. It can also be seen that the coefficient for Employment Deprivation is much smaller. Overall, although Ridge regression was successfully fit in the presence of highly correlated features, model interpretability suffers—the model would have been more easily understandable if Employment Deprivation Rank was excluded and the coefficient for IMD were slightly lower.

In extreme cases, run times may become very long and certain solvers may fail. In our case, training was quick and returned a fitted model with no difficulties. All 66 features have nonzero coefficients. The cross-validation performance (pseudo-R^2) is 0.26. We will use this model's performance as our baseline as it is a simple model to fit and it is the first model that we have fit. We will soon see that this is a reasonable performance compared to other models. While performance is reasonable, model interpretability has suffered to some extent. Highly correlated features with nonzero coefficients are in the model. Sometimes their coefficients have different signs, which means that looking at only one of them does not give the full picture of how that feature affects the model. This point is illustrated in Figure 7.3—the coefficients for IMD and employment have opposite signs, and so their contributions to the linear predictor cancel out to some extent. In other cases, the coefficients of correlated feature have the same sign, but if only one was in the model, then its coefficients would be much larger and it would be more obvious that it was an important feature.

7.4.4. LASSO

In their paper "Regularization and Variable Selection Via the Elastic Net," Zou and Hastie 2005 state that

> *If there is a group of variables among which the pairwise correlations are very high, then the LASSO tends to select only one variable from the group and does not care which one is selected.*

While in the analysis they were writing about, this is actually not ideal (and hence they were motivated to introduce the elastic net); in our case, having LASSO's implicit feature selection choose only one of each group of correlated features in our data seems like a reasonable and useful outcome. We will end up with a model that still performs well, while being easier to understand. (An alternative—which might work better—is to construct a new feature from the correlated features. This is discussed in Section 7.4.5.)

Figure 7.4 shows the output of k-fold cross-validation. The number of features selected as the LASSO penalty increases is shown along the top of the figure. The dashed vertical line close to 59 selected features is at the maximum pseudo-R^2. The dotted line at 31 features is a simpler model whose performance is within one standard deviation of the model with the best cross-validation performance. The performance of this model is a cross-validation pseudo-R^2 of 0.274, which is better than the Ridge regression model above. We would therefore choose this model over the Ridge regression model.

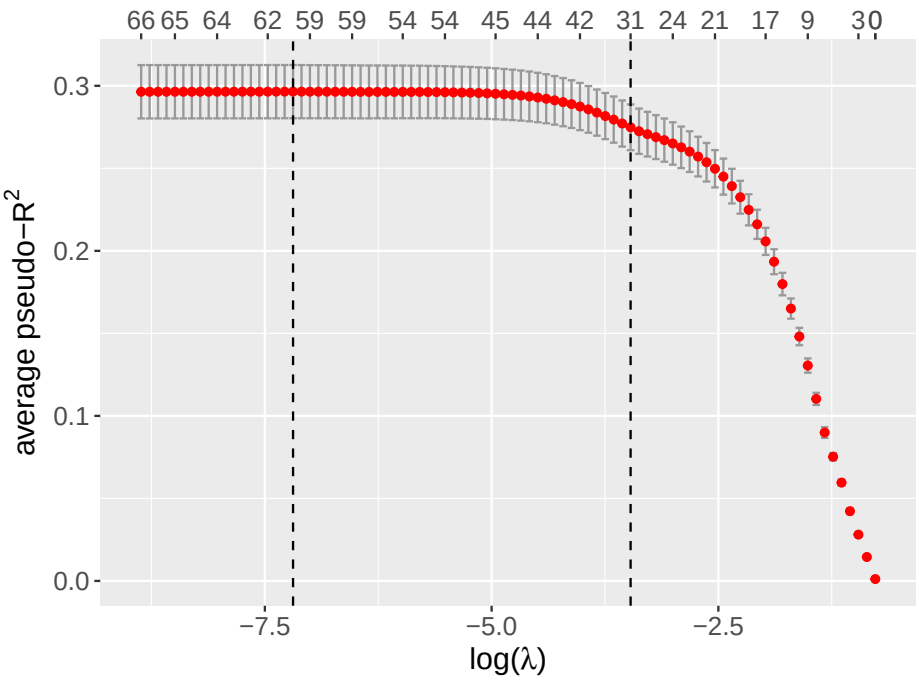

Figure 7.4. stats19 elastic net. α, the mixing parameter between Ridge regression ($\alpha = 0$) and the LASSO ($\alpha = 1$), is set to 0.99**.**

We extracted the coefficients for all features that are 0.70 or more (absolute value) correlated with IMD rank. Table 7.1 shows the correlation of each feature[51] and also the coefficients from the Ridge regression and the LASSO models. It can be seen that the Ridge regression model is complicated—inasmuch as every coefficient is nonzero. In addition, some of the effects are in opposite directions. For example, the features income_rank and as_rank (adult skills subdomain which measures the lack of qualifications in the resident working-age adult population) are both positively correlated with IMD rank, yet their coefficients are of opposite signs. The LASSO model is simpler, with only four features being selected. It will be much easier to discuss and communicate this model to stakeholders. Nevertheless, it is far from perfect. The coefficients for the two last features in the table are both negative, yet their correlations are of opposite signs—which will mean that their overall contribution to the model mostly cancels out.

[51] Definitions are as follows: imd_rank (Index of Multiple Deprivation) is an overall average measure of deprivation. income_rank measures the proportion of the population experiencing deprivation relating to low income. employment_rank measures the proportion of the working-age population in an area involuntarily excluded from the labor market. hd_rank (Health Deprivation and Disability) measures the risk of premature death and the impairment of quality of life through poor physical or mental health. as_rank (Adult Skills) measures the lack of qualifications in the resident working-age adult population. cayp_rank (Children and Young People) measures the attainment of qualifications and associated measures. crime_rank measures the risk of personal and material victimization at local level. c_ts063_occ_9_pct measures the percent of residents aged 16 years and over in "elementary occupations" during the week before the census.

Table 7.1. This table shows the Ridge regression and LASSO coefficients for all features whose (absolute) correlation with IMD rank is 0.70 or more. It can be seen that the Ridge regression model is complicated, with every feature having a coefficient and some of them going in opposite directions. The LASSO model is simpler, with only four features being selected, though even in this simpler model we see the effects going in opposite directions.

feature	correlation	coef_ridge	coef_lasso
imd_rank	1.00	−0.0054	−0.0011
income_rank	0.95	0.0025	0.0000
employment_rank	0.92	0.0004	0.0000
hd_rank	0.84	0.0006	0.0000
est_rank	0.82	0.0007	0.0000
as_rank	0.78	−0.0018	0.0000
cayp_rank	0.76	0.0006	0.0002
c_ts063_occ_9_pct	−0.74	−0.0063	−0.0014
crime_rank	0.71	−0.0012	−0.0014

Overall, the LASSO provides a model with better performance than Ridge regression, and its implicit feature selection tends to select just one (or a small number) out of each group of correlated features. There are some potential practical difficulties though, and these are discussed in Section 7.5.3 below.

Before we move on, we use this opportunity to discuss coefficient paths—the values that each coefficient takes as the penalty increases. In an insurance rating context, two high correlated features can each have a positive impact (say) on risk. When fitted separately in a GLM, each will have a positive coefficient. When both features are entered into the same GLM, though, it is possible that one will have a large positive coefficient and the other will have a negative coefficient. This is known as a reversal. If either of these two features are self-declared by the policyholder, or chosen at the time the policy is taken out, then this reversal is almost guaranteed to cause adverse selection. An example is if the correlation between car value and deductible leads to lower deductibles having higher discounts. Policyholders will obviously choose higher discounts. In this particular case, it might be possible to exclude the deductible from the model, model claims gross of the deductible, and then manually calculate the deductible scale. Such manual intervention is not always possible. The LASSO can sometimes help, i.e., it is possible—but not guaranteed—that as penalization increases, the LASSO will deal with reversals for us. Figure 7.5 shows the full coefficient paths for the highly correlated features shown above in Table 7.1.

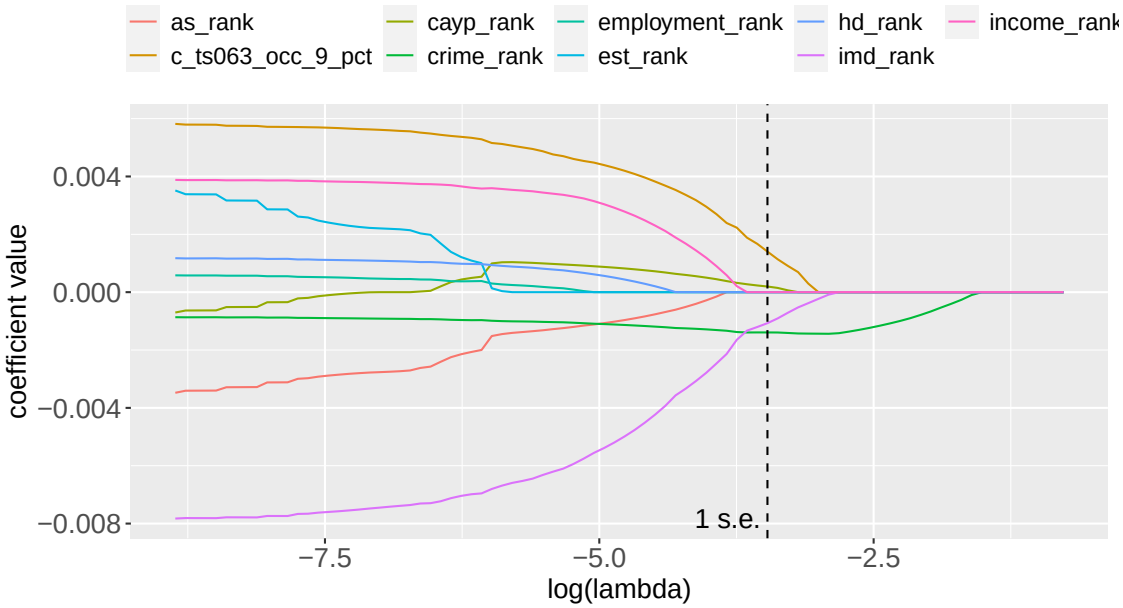

Figure 7.5. Coefficient paths for some highly correlated features. Models with small amounts of penalization show reversals and high coefficients. At the 1 s.e. model, most of the reversals have been dealt with and the coefficients are much smaller.

It can be seen that among models with little penalization (far left), the coefficients were very large, and there we many reversals (given that these features are all picking up the same effect, the coefficients should all have the same sign[52]).

7.4.5. Principal component analysis

In Table 7.1 we saw that the LASSO selected four out of nine highly correlated features. All of these features involve deprivation. The only feature in this list that is not directly related to deprivation is c_ts063_occ_9_pct. This feature is the percentage of households in an area where employment is in an "elementary occupation". A high percentage here may also signify a highly deprived area. The features that have been selected are therefore really a proxy for any measure of deprivation. In communicating this, we cannot really say that the main effect being measured is crime or elementary occupations. It might therefore be easier to replace all of these features with one feature representing deprivation and to have just one coefficient for this replacement feature.

[52] One feature, c_ts063_occ_9_pct, had a negative correlation with the others, but we have multiplied its coefficient by −1.

Principal component analysis (PCA[53]) potentially allows us to do this. It replaces our original features with a new set of features which are not correlated with each other. The new features are linear combinations of the old features. If we are lucky, inspection of the weights of these new features might allow us to describe them in some general and useful way.

The largest loadings of the first principal component from a PCA on our data are as follows:

$$\begin{aligned} \text{pca\_1} = \quad & 0.0275 * \text{imd\_rank} \\ & + 0.0274 * \text{income\_rank} \\ & \dots \\ & - 0.0307 * \text{c\_ts063\_occ\_9\_pct} \end{aligned}$$

The imd_rank and income_rank are indices of deprivation with higher scores meaning less deprived. The positive loadings above mean that high values of the first principal component mean that households in LSOA do not suffer from deprivation. c_ts063_occ_9_pct is the percent of households in the LSOA where employment is in "elementary" occupations. Given the negative loading, higher values of this feature will be associated with low values for this component. Overall, then, this component is very much related to deprivation, with higher values relating to less deprivation.

The second feature (loadings not shown here) is harder to understand. Large positive loadings are given to the features professional occupations and working from home. Large negative loadings are given to the features country of birth in Europe and skilled trades occupations. Further features are also not that easy to understand. Therefore, in our particular case, the PCA components may not help that much with interpretability.

Regarding performance, we note that in GLMs, the linear predictor is a linear combination of features. Therefore, with appropriate coefficients a GLM using PCA based components (which are themselves linear in the original features) can give exactly the same predictions as any GLM based on the original features. In addition, a small number of principal components represent (in a mathematical sense) much of the information conveyed by our features (as measured by variance). In our case, the first two components represent almost 90% of the variance of our data. We can therefore hope that our PCA based model will have the same or similar performance as our original model but with less features.

[53] PCA is a dimensionality-reduction technique that transforms correlated variables into a smaller set of uncorrelated variables called principal components, which capture the most variance in the data. We assume that the reader is familiar with it.

In our case study, performance using only a small number of the principal components is not good. We trained a model using enough components (the first 17) to capture 99% of the variance of the data. Pseudo-R^2 was only 0.13 - this is much worse than previous models. The reason for this (as discussed in Maitra and Yan (2008)) is that PCA is unsupervised - it does not look at the target variable when creating components. There might be a component that is quite minor as measured by the amount variance in the original data that it accounts for, but is very important in predicting accident frequency. That is exactly the case for us. In the LASSO fitted on all the principal components, the largest coefficient is for component 54, despite accounting for less than 0.001 % of the variance in the original data. This situation still seems surprising and is further discussed in Section 7.5.4.

PCA has not led to a simpler or more easily interpretable model. In addition, since each component is made up by loadings on every original feature, a model based on PCA requires all of the features and so is actually harder to implement operationally than the simple LASSO model which did not require all of them. Operational issues are not always a consideration, though. We might be interested in ranking areas in terms of accident risk as part of a broader geographical risk analysis. As in the case of our FAA-NSTB study, we might implement our findings by assigning each zip code to a group and so how we arrived at these groups does not need to be operationalized. In these cases, the fact the PCA requires all of the original features is not a disadvantage, and as long at it provides other advantages (such as making the models easier to fit) we should use it.

Before the LASSO was easily available, PCA would have been (and still is) one way to get a non-penalized linear model to converge and to have sensible confidence intervals for parameter estimates. Overall, as we have seen PCA has advantages (e.g., dimension reduction, interpretability of components, non-correlated features, and so quicker run times, convergence of non-penalized GLMs) and possible disadvantages (e.g., it is not supervised so early components may not be predictive, use of all original features is necessary). Whether or not PCA is overall beneficial will depend on the particular analysis and more generally on the domain (e.g., insurance, genomics, chemical engineering, etc.).

Other than PCA, any other dimension reduction technique can also be used to try to deal with highly correlated features. Two examples are Independent Component Analysis (ICA) and autoencoders. ICA transforms the features to be independent. Unlike PCA there is no focus on trying to represent as much as possible of the variation of the original data in the first few features. We carried out ICA followed by the LASSO on our data. Performance was the same as for the LASSO and the components were of interest. Autoencoders use neural networks to create low-dimensional (nonlinear) representations of our data, and they are not discussed further here.

The above dimension reduction techniques are all unsupervised, and, as discussed, there is no guarantee that the components extracted will be important in terms of predictive power. Partial Least Squares (PLS, see An Introduction to Statistical Learning James et al. (2021), Section 6.3) is a supervised method that creates components in such a way that they are important for prediction. It was specifically created for situations like ours, where many of the features are highly correlated. PLS has been extended to work in the GLM setting and is called, unsurprisingly, PLS-GLM.[54] To ensure sparsity, the PLS-GLM components can be extracted and then used as input to the LASSO. In our case, the LASSO selects the first 10 PLS-GLM components and performance is reasonable. The components themselves are also fairly interesting. We don't discuss PLS-GLM further in this note, but it certainly seems useful, especially for gaining insight into the underlying features that are driving performance of the model.

Cluster analysis is often discussed alongside dimension reduction – because they are both examples of unsupervised learning. However, in our context, it does not seem particularly useful. We could create clusters of similar areas based on their features. This might be of interest from a data exploration perspective - but this would not necessarily lead to useful features to use in a regression.[55]

7.5. Practically Speaking

7.5.1. Fold variable in time series data

An observation for each LSOA is present for every quarter from 2018 to 2022. Consider the case where, within each LSOA, observed accident frequency is fairly stable over time. If we choose our fold variable for each line in our dataset at random, then the same LSOA will be in different folds. If we then see good cross-validation performance, we will not be sure if some of the performance is the result of the model having seen the LSOAs in the validation dataset during training. We therefore allocated all the observations for each LSOA into one (randomly selected) fold. In private auto insurance, the same policy number can be present in the dataset many times. For example, in any one policy year, each time there is a midterm adjustment to the policy (for example, a change of address) a new line of data may be generated. And the same policy number may be present over multiple years if the policy renews. For the same reason as discussed in our case study, it seems sensible to allocate all observations for any one policy to a (random) fold.

[54] At the time of writing This is available as a the plsRglm package in R, see Bertrand and Meyer (2018).

[55] Incidentally the distance that each observation is from the cluster centroid can be useful, for example in fraud modeling. In some domains, it can be the case that customers or transactions who are unusual in some way are more likely to be fraudulent. Isolation forests are an easy machine learning technique to find such customers or transactions.

7.5.2. Hinges and splines with correlated features

The astute reader may have noticed that the graphs and tables of coefficients in this chapter were only for the original features. No hinges or splines were added. In the authors' experience, it is often the case that the contribution to predicted accident frequency of this type of data is not a linear relationship. One possibility is that across much of the range of a feature there is no effect and only at the extreme values is there an increase in risk. Our reason for excluding splines here was purely for simplicity of exposition. If splines are included at the start of the analysis before highly correlated features are excluded, then things get a little more complicated. This is because it is possible that feature selection will select one of the original features for the main effect and the spline of one of the other features to capture some of the nonlinear shape of the effect. Adding this element to the case study was not needed to illustrate the main points in this chapter.

7.5.3. LASSO in the presence of highly correlated features

A difficulty encountered in this case study was that, in the presence of highly correlated features, the LASSO implemented in one software often failed to converge and the run times were very long.[56] Part of the solution to this was to use the elastic net with high values ($\alpha = 0.95$). Proprietary software with custom solvers can perform more consistently. In either case, it does seem reasonable to use common sense first and remove the most highly correlated features.

7.5.4. High coefficients in later principal components

Despite principal component analysis (PCA) being unsupervised and, therefore, us not being totally surprised that a later component might be predictive, the extent in our study is unusual. Investigation showed that we had made a basic error in our features. We had originally noticed that one of the deprivation indices (living environment) is based partly on air quality and partly on the accident rate itself. We therefore only included the air quality feature. We excluded the living environment and accident rates features as it is clearly an error—and the ultimate in leakage—to include a feature derived from accidents to predict the accident rate. However, we did still include the overall IMD.

[56] The ability of the model to converge at all, regardless of runtime, also depended on the hardware on which it was run

Principal component 54 has the following main loadings:

$$\begin{aligned} \text{pca\_54} = &+ 0.274 * \text{imd\_rank} \\ &- 0.109 * \text{income\_rank} \\ &- 0.055 * \text{bhs\_rank} \\ &- 0.054 * \text{hd\_rank} \\ &- 0.053 * \text{employment\_rank} \\ &- 0.038 * \text{crime\_rank} \end{aligned}$$

We can see that this feature "reverse engineers" the living environment feature by deducting the other deprivation indices from the overall index. Hence, one feature in our model is based partly on accident frequency itself, which, of course, is simply wrong. The way to proceed at this stage would be to remove the overall IMD and to repeat the whole analysis. (The authors leave this to the interested reader.)

Therefore, while PCA has not been very helpful to us in terms of interpretability or sparsity of our model, it has provided us with an interesting way to check the quality of our data preparation. It seems reasonable that whenever regression is carried out on principle components and large coefficients are fitted to the later components, an investigation should be carried out to understand why and to ensure there are no data preparation issues.

8. Modeling in Two Stages

Often a single GLM is not sufficient to create a full rating plan, and thus there are situations where we want to do something along the lines of:

- Fit a GLM or insist on some fixed starting point as a first stage.
- Then carry out some adjustment or another modeling process as a second stage.
- Finally, recombine the original GLM and the results of the second stage into a model which can make predictions based on both stages.

There are various situations when splitting out one aspect of our model into a second stage has operational or technical advantages. Some examples are:

- Sometimes, when we submit a new rating plan to a regulator, we don't want too much to change at once. One easy way to ensure control over what changes is to somehow insist that our existing rates are a fixed first-stage model and to fit a second-stage GLM to see what incremental changes we might want to propose.
- We have one or more features for which the relativities are fixed, and we want to fit a GLM given these loads/discounts. An example often seen in practice is a deductible discount scale.
- In a rating system, we have some basic features, such as age and location, provided by the customer. We have further data enrichment features that are available at the time of quote (at a cost). The enrichment features are correlated with the basic features. An example might be that the enrichment feature is credit score (if it is permitted), and this is correlated with customer age. Consider the case where the average load in the rating plan for a 20-year-old driver consists of a 20% load based on age and a further 5% load based on credit score (because average credit scores are poor for younger drivers). If the enrichment services which provided credit score data fail for a day (say), then the average quote on that day will be 5% too cheap for those drivers. In such a case, it might be beneficial to first fit a GLM for customer age, and as a second stage add credit score to the model while insisting that the relativity for age does not change ("controlling" for age). This way, if enrichment fails at the time of the quote, at least the age relativity

will be correct.[57]

- Sometimes, a GLM is not sufficient for our needs because it cannot fully capture the relationship (we will explain this further below). Consider dealing with risk which varies by geographical area. It is not easy to deal with this purely in a GLM framework. We might therefore want to fit all the features that we can within a GLM. Then, as a second stage, we can complete our geographical area rating using some other approach.

In this chapter, we will discuss some of the reasons for two-stage modeling in more detail and how technically we might go about it. We will look at some aspects of our simulated auto insurance and FAA-NTSB case studies for which we might want to use this technique. There are three straightforward practical techniques we will need:

- fixing coefficients with an offset
- using the residuals from a first model in a second model
- combining parts of multiplicative models.

We will not exhaustively cover all situations where these might be used, but once the three ideas are understood, their application to novel modeling challenges is straightforward.

8.1. Fixing Coefficients With an Offset

Offsets and reasons for using them are discussed in Goldburd et al. (2025) Section 2.6 and in significant detail in the clear and easy-to-read paper, "Applications of the Offset in Property-Casualty Predictive Modeling" (Yan et al. (2009)). The latter source covers much of what we discuss in this chapter. The mathematics of offsets and their application was further extended in Shi (2010).

What follows is an example to motivate why we might want to use an offset and how to do it.

Consider a very simple model in private passenger auto. There are only two features, age and no claims bonus (NCB). NCB is a number which is awarded to each insured based on their claims history. Every driver starts with an NCB of zero. If they drive for a certain number of years without any claim, then their NCB increases by one up to a maximum of four. Age will be represented by a categorical variable A, B, ..., E, where A represents the lowest ages and E represents the highest ages.[58]

[57] As an aside, another way to deal with this is to include in the credit score table in the rating plan a line for failed enrichment, which varies by age.

[58] In practice we would never replace a numeric feature such as age with a categorical feature, as it destroys relevant information about the order and proximity of the ages. We have done it here for pedagogical reasons only.

Data were simulated to represent this situation, and a sample of the simulated data is shown in Table 8.1. Age group takes values A, B, ..., E and NCB takes values 0, 1, ..., 4. gt_rel_ncb (the true relativitiy for NCB) and rel_ncb_mkt (the relativity for NCB communicated by the marketing department) will be discussed below. num_cl is the simulated number of claims.

Table 8.1. Simulated data for private auto example. Age group takes values A, B, ..., E. NCB takes values 0, 1, ..., 4. gt_rel_ncb is the true relativity for NCB, and rel_ncb_mkt is the relativity for NCB communicated by the marketing department. num_cl is the simulated number of claims.

id	age	ncb	gt_rel_ncb	rel_ncb_mkt	num_cl
1	B	1	0.95	0.65	0
2	D	3	0.85	0.40	0
3	D	3	0.85	0.40	0
4	A	0	1.00	1.00	0
5	D	2	0.90	0.50	1
6	B	1	0.95	0.65	0
7	D	4	0.80	0.20	0
8	D	2	0.90	0.50	0
9	D	3	0.85	0.40	1
10	A	2	0.90	0.50	0

Typically, only drivers at older ages have been insured long enough to earn the higher levels of NCB, and so there is a very strong relationship between age and NCB (see Table 8.2).

Table 8.2. The number of insureds in each combination of age and NCB in the simulated data. There is a strong relationship between age and NCB. (Note that the age categories and NCB step-back rules were not specifically defined in the simulations—we simply ensured that the typical expected correlations between older ages and higher NCB levels exist.

	NCB 0	1	2	3	4
age					
A	132381	54519	12194	1237	26
B	55624	82213	49354	12283	952
C	11053	49903	77029	50248	11291
D	920	12187	49027	81946	55690
E	22	1178	12396	54286	132041

Since the data are simulated, we know the underlying ground truth for the true relativities for age and NCB (see Table 8.3).

Table 8.3. The ground truth relativities for NCB and age used to simulate the data.

age	A	B	C	D	E
true relativity	1	0.7	0.6	0.5	0.4
NCB	0	1	2	3	4
true relativity	1	0.95	0.9	0.85	0.8

For simulations, we set claim frequency for age A and NCB 0 to 10%, and then the frequencies for all other combinations of age and NCB can then be derived from Table 8.3.

For simplicity, we assume that if there is a claim, it will be for exactly $10,000. This means that the correct risk premium for an insured in age band A and NCB 0 is the product of the intercept of the claims frequency model (0.10), the relativity for age group A (1.0), the relativity for NCB level 0 (1.0), and the expected severity (10,000):

$$0.10 \times 1.0 \times 1.0 \times 10000 = \$1,000$$

Likewise the correct risk premium for an insured in age band E and NCB 4 is

$$0.10 \times 0.4 \times 0.8 \times 10000 = \$320.$$

A Poisson GLM with a log link was fitted to the data, using age and NCB as predictors. The exponents of the fit coefficients ($\exp \beta$) from the GLM are close to the ground truth and are shown in Table 8.4.

Table 8.4. $\exp \beta$ are the exponents of the fitted coefficients from a GLM with age and NCB. It can be seen that they are close to the true relativities.

age	A	B	C	D	E
true relativity	1	0.7	0.6	0.5	0.4
$\exp \beta$	1	0.702	0.605	0.494	0.390
NCB	0	1	2	3	4
true relativity	1	0.95	0.9	0.85	0.8
$\exp \beta$	1	0.937	0.900	0.846	0.826

Based on the coefficients from the fit GLM (and given that the exponent of the fit intercept is 0.10), the risk premium for an insured in age band E and NCB 4 is

$$0.10 \times 0.39 \times 0.826 \times 10000 = \$322,$$

which is close to $320, which we know to be correct.

When we present our relativities to the business, we are told that they cannot use the NCB relativities. This is because the marketing department has already communicated to potential insureds a scale of discounts for NCB which are much larger than those output by the GLM. Table 8.5 shows the discounts communicated by the marketing department[59]—it can be seen that they are far larger than those from the fitted GLM or the ground truth.

Table 8.5. $\exp\beta$ are the exponents of the fitted coefficients from a GLM with age and NCB. The table compares these to the marketing relativities (or discounts). We can see that there are some large differences.

NCB	0	1	2	3	4
true relativity	1	0.95	0.9	0.85	0.8
$\exp\beta$	1	0.937	0.900	0.846	0.826
marketing relativity	1	0.65	0.50	0.40	0.20

If we use the GLM relativities for age and the marketing discount for NCB, then the risk premium for an insured in age band E and NCB 4 is

$$0.10 \times 0.39 \times 0.20 \times 10000 = \$78,$$

which is far too low.

Since age is highly correlated with NCB, we should be able to fix this at least partially by adjusting the coefficients for age to allow for the very large marketing discount for NCB level 4. We could try to do this manually, but we can actually achieve this very easily by just "telling" the GLM that it must use the marketing relativities for NCB and only to fit the coefficient for age.

Given a log link, we use μ to refer to the expected (annual) claim frequency[60] and $\eta = \log(\mu)$ to refer to the linear predictor (i.e., the log of the expected claim frequency). Then for an insured in age band E and NCB band 4, our first model was

$$\begin{aligned}\mu = {} & \text{intercept} \times (\text{coefficient for age band E}) \\ & \times (\text{coefficient for NCB band 4})\end{aligned}$$

[59] This is a very extreme example for illustrative purposes.

[60] In our simplified simulation example, every observation represents exactly one year of exposure. In most practical situations, each observation represents an amount of exposure (ex) between zero and one, the claim frequency is calculated as num_cl/ex, and the expected number of claims for an observation is $ex \times \mu$.

and taking logs,

$$\begin{aligned}\eta &= \log \mu \\ &= \log \text{intercept} + \log(\text{coefficient for age band E}) \\ &\quad + \log(\text{coefficient for NCB band 4}).\end{aligned}$$

Given the marketing discounts for NCB, we now need to fit a model where the linear predictor is given by

$$\eta = \log \mu = \log \text{intercept} + \log(\text{coefficient for age band E}) + \log(0.20)$$

The value log (0.20) in the above model is known as the offset because it adjusts (offsets) the linear predictor by a given amount.

A Poisson GLM with a log link was fitted to the data, using age as the predictor and including an offset for the logarithm of the relative marketing discount. The coefficients for age in this GLM are shown in Table 8.6.

Table 8.6. The exponents of the coefficients for age from a GLM where the marketing relativities for NCB are offset. The fitted intercept (not shown in the table) is 0.112.

age	A	B	C	D	E
true relativity	1	0.7	0.6	0.5	0.4
exp β	1	0.702	0.605	0.494	0.390
exp β when NCB offset	1	0.849	0.925	0.980	1.057

Given an estimated intercept of 0.112 and using the marketing discounts for NCB and the revised relativities for age, the risk premium for an insured in age band E and NCB 4 is

$$0.112 \times 1.057 \times 0.20 \times 10000 = \$237,$$

which while still far from the true risk premium of $320 (or our original risk premium of $322) is significantly better than $78.

In this case, the GLM has used the strong relationship between age and NCB to correct for the excessive NCB discount communicated by the marketing department. For example, for NCB group 4 the true relativity is 0.8 and the marketing discount is 0.2—hence without adjusting the age coefficient—the resulting premium will be 0.25 ($\frac{0.2}{0.8}$) of the correct amount. Since most of the NCB group 4 is in age group E, the increase of the age coefficient for age group E to 1.057, compared to 0.39 in the original model, helps to make up some of the difference.

To the extent that other features are correlated with the feature to which the offset applies, their coefficients will change in order to bring the model predictions closer to what they would have been using all the features. We end up charging less unreasonable premiums to all customers despite our marketing department's bold promises on NCB discount.

In the introduction to this chapter, we mentioned that in practice we may wish to offset a given deductible relativity. In this and similar cases, if we are fitting both frequency and severity models, the question arises: how much of the discount should be offset in the frequency model and how much in the severity model? Unless it is obvious from the situation that the effect should be offset totally in one or the other, a simple approach is to offset equally (i.e., the square root of the load/discount) in each.[61]

Before moving on, we note that Table 8.6 exhibits what is known as a "reversal": the estimated coefficients for age increase with driver age, even though we know they should decrease. While the application of the offset has indeed left us with less unreasonable premiums, the outcome is far from perfect. For example young, safe drivers will now be grossly undercharged.

8.2. The Hypothesis Space of GLMs

In the introduction to this chapter, we mentioned that the hypothesis space of GLMs is occasionally insufficient for our needs, and so the GLM cannot "grasp" some relationships between risk classes. Here we discuss this point in more detail. In the two following sections we show how we can use offsets or residuals to create two-stage models, as well as the mathematical equivalence of these methods.

Once a GLM has been fit, we essentially have a function which maps from the input features to the predicted target. If $x_1, x_2, \ldots, x_p$ are the features and $\hat{y}$ is the predicted target, then our GLM is a function:

$$f : (x_1, x_2, \ldots, x_p) \mapsto \hat{y}.$$

For example, if we fit a GLM with an identity link, and the only feature is aircraft age and the estimated coefficient is 2, then the fit GLM is

$$f : (x_1) \mapsto 2 \times x_1,$$

[61] Alternatively, more accurate approaches could be used. For example, we could fit separate frequency and severity models and include the feature that we want to offset. We can then inspect the coefficients fit in the frequency and severity models and use them to inform how we split the offset.

where x_1 is aircraft age. The estimated function is known as a hypothesis, and the set of all functions that could have been fit is known as the hypothesis space. There are an infinite number of possible functions in this GLM's hypothesis space because there are an infinite number of coefficients that could be chosen instead of 2. Despite there being infinite functions in the GLM hypothesis space, there are a great many that are not. For example,

$$f : (x_1) \mapsto 2 \times \frac{1}{x_1}$$

is not (because with an identity link, μ is not a linear function of x_1). We can get around this quite easily by creating a new feature $x_2 = \frac{1}{x_1}$ and then we would have

$$f : (x_1, x_2) \mapsto 2 \times x_2.$$

Now consider a harder case from auto insurance. Each car is associated with two locations: the house where the owner lives (x_{1long} and x_{1lat}) and the place where the car is parked overnight (x_{2long} and x_{2lat}). We might want to test whether the distance between these two locations impacts the probability of an accident or of a theft. Using the two sets of longitude and latitude as features in a GLM will not help. Rather, we need to create a new feature to represent the distance:

$$x_3 = \sqrt{(x_{1long} - x_{2long})^2 + (x_{1lat} - x_{2lat})^2}$$

and use this new feature in the GLM. This idea of engineering new features is exactly how we overcame the two limitations of the GLM hypothesis space discussed so far in this note. We engineered hinge and step functions to overcome the GLM hypothesis space only containing functions which are (roughly) linear combinations of the features. Allowing the GLM to fit interactions also involves engineering new features (although this is often built into the software which fits the GLMs). We have to pay very careful attention to feature engineering to successfully use the GLM with its relatively simple hypotheses space.

An alternative to GLMs is to use a model with a less restrictive hypothesis space, allowing us to avoid much manual feature engineering. Back in 1989 it was proved (Cybenko (1989), Universal Approximation Theorem) that, given our original features, and without us doing any feature engineering, a neural network can approximate any continuous function of those features as accurately as we would like. For example, with appropriate training the neural network will find the correct shape for the nonlinear effects. Other machine learning methods, such as gradient boosting machines, given the original features, have a different hypothesis space than GLMs and might be able to fit a model

which performs better than a GLM.

The downside of models that are more complex than GLMs is that they are not easily interpreted. If a customer or a regulator asks for an explanation of the rate, it will not be easy to provide it. There is a compromise. We can use a GLM for the features that it can easily fit and for those which we can easily engineer. Then, as a second stage, we can feed the residuals from this model into a more complex model to deal with aspects of the modeling task which are not as easily represented or discovered by a GLM. We will then need to find a way to combine the output from the more complex model, or the features that it has generated, with the GLM.

A particular widespread (in current practice) application of the above is geographic rating. Consider zip codes (or census tracts), an extreme case of an HCCV. We can partially deal with this by not modeling zip codes directly but rather adding features related to zip codes to our dataset, such as population density or the distance to the nearest school or police station or fire hydrant. However, whether or not we have such data, there is another point to consider. Following Tobler (1970) the famous (and maybe obvious) first law of geography is "Everything is related to everything else, but near things are more related than distant things." In our case, we expect that claims experience in nearby zip codes is similar. Although we don't have a lot of information in each zip code, if we can somehow smooth or average the claims experience of nearby zip codes, this might be an important feature in our model. It is not easy within a GLM framework to carry out geospatial smoothing.[62] This suggests to us again a two-stage model:

- Fit a GLM to all features except zip code (and possibly zip code related features).
- Use the output of the GLM (offsets or residuals) to carry out geospatial smoothing with a different model type.
- If necessary, combine the results of the geospatial smoothing with the GLM.

8.3. Two-Stage Models Using Offsets

In this section, we show how we can use an offset to fit another model which aims to find additional relationships between features and risk, above and beyond what a first-stage model has found. The example we will use is our FAA-NTSB case study, and we will look to fit a second-stage model to produce a smoothed view of geospatial risk across the US. We use the zip code of the person to whom the aircraft is registered as the location of interest. (This example is for pedagogical interest only—there is no particular reason to assume that the zip code of the registered owner should impact the frequency of the types of aircraft accidents in the NTSB database.)

[62] It can be done to some extent with a bespoke penalization scheme. See for example GLUM Contributors (n.d.).

The model fit so far as a result of our work in previous chapters is our first-stage model. We have a file, which for each row in our original data, contains a unique id and the predicted[63] number of claims from stage 1. To this file, we add the zip code and then look up the longitude and latitude. A sample of the resulting file is shown in Table 8.7. The "ex" column is the number of aircraft-years represented by each line—for these observations, 1.[64]

Table 8.7. A sample of the records and predictions from our first-stage model. (Note that exposure is one (year) on these records, but this is not true across all observations.)

						number of claims	
unique_id_line	fold	zip5	ex	lon	lat	actual	expected
285985	5	15668	1	−79.670	40.461	0	0.00085
1670670	0	21822	1	−75.636	38.281	0	0.00260
1769520	0	60077	1	−87.757	42.035	0	0.00932
454832	2	83607	1	−116.751	43.709	0	0.00145
1387894	4	85248	1	−111.870	33.215	0	0.00183

For convenience, we summarize this file to zip code level. (It is not obvious that we can do this without changing the results of our second-stage model—but in the case where we use a Poisson distribution, it happens to be true.) The resulting table has information on about 700,000 aircraft-years spread across 20,000 zip codes. For this example, we use the first five digits of zip code, we exclude Alaska and Hawaii, and also only use records for registered owners who are individuals (as opposed to corporations). A sample of the resulting table is shown in Table 8.8.

In order to visualize any possible geographic patterns, we now plot the residuals from the first model on a map. Given the multiplicative nature of all our modeling so far, we define "residual" as the actual number of claims divided by the predictions from the stage 1 model ($\frac{y_j}{\hat{y}_j}$, where j represents a zip code). (Where the actual number of claims is zero, we replace the zero residual with a small number, e.g., 0.01—this will be important later on, where we might want to take the log of the residuals.) Figure 8.1 shows the log of the residuals summarized to the first three digits of the zip code and where we include only zip codes with more than 500 aircraft-years.

[63] Sometimes referred to as expected.
[64] The techniques being discussed will work for any (positive) numeric values for exposure.

Table 8.8. A sample of the records and predictions from our first-stage model summarized to zip code level (first five digits). For example, across all our records for zip code 15668, the sum of aircraft exposure years is 15.65, the total number of claims is zero, and the sum over these records of exposure times predicted (annual) frequency from our previous model is 0.014.

				number of claims	
zip5	ex	lon	lat	actual	expected
15668	15.65	−79.670	40.461	0	0.014
21822	15.14	−75.636	38.281	0	0.037
60077	9.37	−87.757	42.035	0	0.039
83607	182.28	−116.751	43.709	1	0.759
85248	249.62	−111.870	33.215	0	0.739

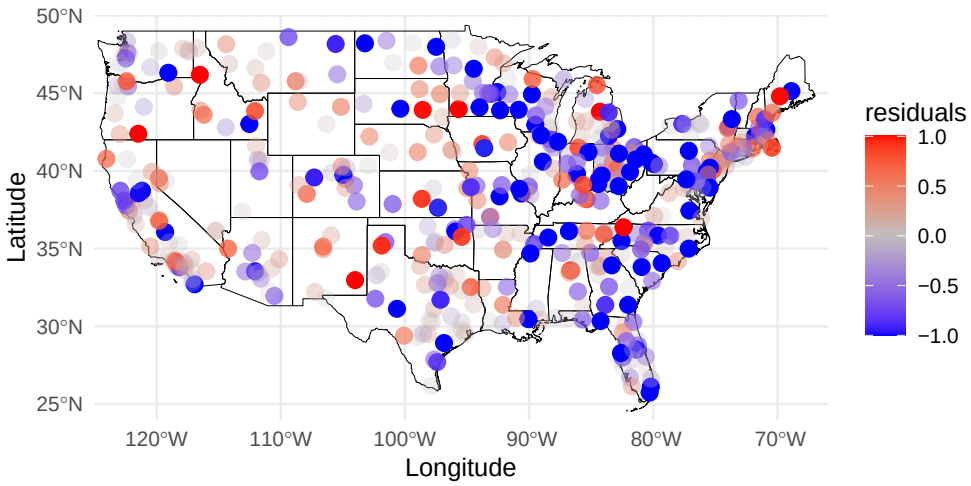

Figure 8.1. The log of the residuals (actual number of claims divided by predictions) from the stage 1 model. Only zip codes (first three digits used here) with 500 or more aircraft-years are shown. Patterns are not easy to spot, but possibly Florida has consistently low residuals, whereas there seems to be a band of higher residuals running from Oregon through Idaho and into western Montana.

We want our second-stage model to predict the number of claims for each aircraft using longitude and latitude of its owner's residence. We also want it to start where the first model left off—a high number of claims for a particular area does not mean that area is a high-risk geography—we might have been expecting a high number of claims simply given the predictions from our stage 1 model. If $pred_1$ and $pred_2$ are the predicted

frequency from our first-stage and second-stage models, we need

$$pred_2 = pred_1 \times \text{geographic effect.}$$

Taking the log of both sides, we have

$$\log pred_2 = \log pred_1 + \log \text{ (geographic effect).}$$

So, if our second-stage model is multiplicative and allows for the use of an offset, then our target variable is claim frequency (with exposure as weights), and we use the log of the expected claim frequency from the first-stage model as the offset.[65] The model type that we chose to use for the spatial smoothing (generalized additive models, GAMs) does allow this formulation.

Our main aim here is to demonstrate the use of an offset to achieve second-stage models, but nevertheless, out of interest we show the result of the spatial smoothing in Figure 8.2.

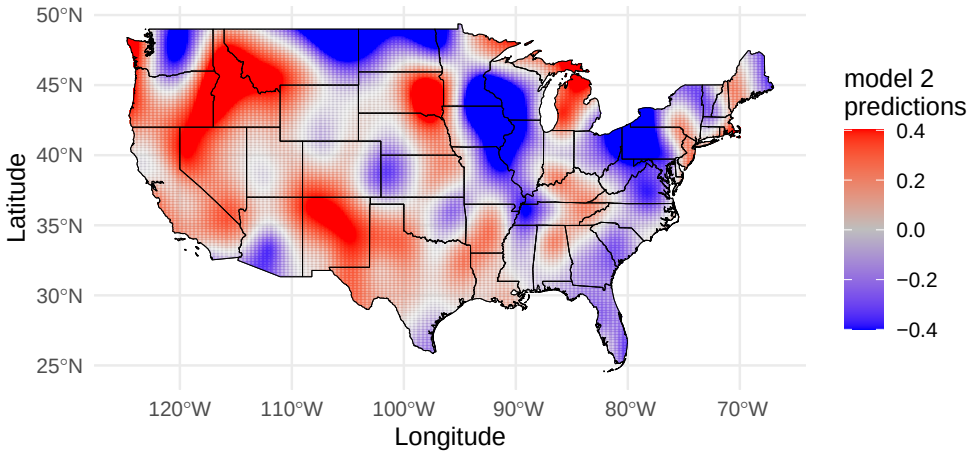

Figure 8.2. The log of the predictions from the GAM are shown. It can be seen that these are high from Oregon through Idaho and into western Montana and somewhat low in Florida.

Above, we used predictions from a first-stage model in a second-stage model. If we are not careful, the predictions of the first-stage model for each observation will be based on having seen the value of the target variable for that observation. The problems this might cause, and how to avoid them, are discussed further in Section 8.9.

[65] An alternative formulation is: Target variable–number of claims; offset–log of (exposure times the expected claim frequency from the first-stage model); weight–not needed.

8.4. Two-Stage Models Using Residuals

When we use a second-stage model as part of our modeling process, there is no guarantee that it will have the option for an offset. In the cases where an offset is not available, we can model the residuals instead. Given the multiplicative nature of our modeling, we define the residuals as the target for the first-stage model divided by the predictions from the first-stage model,[66] and we use these as the target for the second-stage model. Continuing to look at our data aggregated to zip code level in Table 8.8, the target for the first row will be $\frac{0}{0.014} = 0$ and for the fourth row will be $\frac{1}{0.739} = 1.353$. Some residuals are based on more data, and we are therefore more certain about them (they have a lower variance). We allow for this in the second-stage model by using the weight option and setting it equal to exposure times the predicted (annual) frequency from the first-stage model.

In our case, the results from the second-stage model with this approach are identical to those when using an offset. We would typically expect this when using a Poisson, log-link model, since it has been shown by Yan et al. (2009) (Appendix A), and then generalized by Shi (2010), that the two approaches are mathematically identical.

8.5. Two-Stage Models Using Preadjustment

This section draws directly from Shi (2010). In order that the interested reader can refer to that paper without confusion, we will use a similar[67] notation here; for observation i, c_i is the actual number of claims, l_i is the cost of claims incurred, and e_i is exposure. Finally u_i is used to refer to the predictions from the first-stage model.

Above, we talked about modeling residuals. For a frequency model, in our first-stage model, the target was claims frequency ($\frac{c_i}{e_i}$), the weights were the exposure (e_i), the predicted frequency is denoted by u_i, and the predicted number of claims is $e_i \times u_i$. We then defined residuals as the actual number of claims divided by the predicted number of claims

$$\frac{c_i}{e_i \times u_i}$$

and we used this residual as a target variable, and we claimed (without much proof) that we should use the predicted claim counts ($e_i \times u_i$) as weights.

If we express our residual as

$$\frac{c_i/e_i}{u_i}$$

[66] Using division rather than subtraction to arrive at the residuals is not a theoretical nicety; it is a practical necessity. This is because the result of this analysis will be an extra table in our rating structure whose relativities will multiply the results until that point.

[67] Our notation is similar but not identical.

we can consider our target as the frequency, adjusted by the predictions from the first model. Where the prediction is high, we reduce the frequency as we don't need the second-stage model to pick up this effect which is already picked up by the first model.

Shi (2010) proved that modeling the adjusted frequency and using weights $e_i \times u_i$ is correct and identical to using the offset. Consistent with referring to the target as "adjusted frequency," Shi refers to $e_i \times u_i$ (which are our claim count predictions from the first-stage model) as "adjusted exposures."

In Table 8.8, we summarized our residuals by each level of the zip code (zip5). Shi (2010) proves that in this case, too, the weight that we used (the sum of the predicted claims for the zip code, $\sum_{\text{i in a given zip code}}(e_i \times u_i)$) is correct.

The case studies in this monograph focus on modeling claims frequency. The techniques discussed (dealing with nonlinear effects, HCCVs, etc.) apply equally to modeling claim severity ($\frac{l_i}{c_i}$) or (annualized) loss cost ($\frac{l_i}{e_i}$). In our discussion here, too, we may wish to see how predicted claims severity from a first-stage Gamma GLM should be adjusted by zip code or how predicted (annualized) loss costs (i.e. risk premiums) from a Tweedie GLM should be adjusted. The process is the same, fit a first-stage model without zip code, summarize the predictions by zip code, calculate the residuals (or "adjusted claim severity" and "adjusted loss cost"), and then fit the stage-two model with appropriate weights.

Shi (2010) provides the correct weights in these situations, too. For the convenience of the reader, we summarize Shi's results (see Shi (2010) , Table 3) in Table 8.9.

Table 8.9. The correct weights for modeling adjusted frequency, severity, or loss costs. The notation is described in the main text. Note in particular that u_i on each line in this table represents the predictions from the first-stage model fitted in that case. The sums are over all observations in a given zip code. If there are 100,000 zip codes, then the resulting dataset will have 100,000 lines, one for each zip code, each line containing the value of the target variable and the weight, as well as any features used in the second-stage model. If the second-stage model requires cross-validation or similar, the summations are per zip code and fold.

Model	Distribution	Response Variable	Weight Variable
Frequency	Poisson	$\sum c_i / \sum(e_i u_i)$	$\sum e_i u_i$
Severity	Gamma	$\sum l_i / \sum(c_i u_i)$	$\sum c_i$
Loss cost	Tweedie(p)	$\sum l_i / \sum(e_i u_i)$	$\sum e_i u_i^{2-p}$

8.6. Recombining the Two Models

The result of the second-stage model fitted above, whether using a GAM or any other method to deal with spatial data, is a file which contains a relativity for each zip code. For any individual aircraft we look up the zip code relativity and apply it to the result of the rating plan so far. Some examples of the result of our analysis are shown in Table 8.10.

Table 8.10. A sample of zip code relativities from the stage 2 model. If the predicted frequency for an aircraft in zip code 15668 from the stage 1 model (i.e., rating plan excluding this table) is 0.001, then the prediction including this table is 0.001×0.53113.

zip5	stage2_pred
15668	0.53113
21822	1.01655
60077	0.92112
83607	1.72816
85248	0.71129

To find a predicted frequency from our two models, we simply calculate a prediction from the first model and multiply it by the zip code relativity from the second model. No further analysis needs to be done.

A practical point to note is that it might be more convenient to provide the results as an allocation of each zip code to an area group—this is discussed further in Section 8.9.2.

8.7. Combining Parts of Multiplicative Models

Our FAA-NTSB model has various features that are related to the make and model of the aircraft. Some examples of these features are:

- faa_eng_hp_char—horsepower for piston engines. Coded as a letter (e.g., "A" represents 0–149 hp and "B" represents 150–249 hp).[68] For example, the Lycoming O-320 series of engines, with around 160 hp, is used in the Cessna 172M—a very popular training aircraft—and is coded as "B."

[68] In reading details of this and the following features, the reader could feel, rightly, shocked. It looks like the authors have taken a useful numeric variable, split it into discrete chunks, and labeled those chunks with letters which have no inherent order. Discarding order and proximity information in numeric data by encoding it as categorical destroys information! However, the FAA data for these fields was not provided as continuous numeric variables, but rather in groups, e.g., for weight, four classes were provided, CLASS 1: Up to 12,499 lb., etc. In addition, we have no a priori reason to assume that there is any order in how accident frequency varies by the weight category.

- faa_eng_thrust_char—thrust for turbine engines, measured in pounds. Coded as a letter ("A" represents thrust up to 10,000 lb.). Lycoming O-320 is not a turbine engine, and its power is measured in horsepower (hp), not pounds of thrust. In the raw FAA data it is coded as thrust 0. In our data preparation we code this engine to "Y," where the "Y" category is reserved for nonturbine engines.
- faa_acft_ac_weight—aircraft maximum gross takeoff weight in pounds, grouped into four classes. For example, class 1 is "up to 12,499 lb.," and class 4 is "UAV (unmanned aerial vehicle) up to 55 lb." The Cessna 172M maximum gross takeoff weight is 2,550 lb., and so it is coded to "A."
- faa_acft_type_acft—fixed-wing single engine (coded to "4"), fixed-wing multi-engine (coded to "5"), and rotorcraft (coded to "6").
- faa_acft_num_eng—the number of engines (one, two, three, or four). The Cessa 172M has one engine.
- faa_acft_num_seats—the number of seats. UAVs have zero seats, other aircraft have from one to 400+ seats. The Cessna 172M has four seats.

In addition to the above, we also have a separate relativity for each make-model of aircraft which we calculated in the chapter on HCCVs. There will therefore be seven tables which combine to give a relativity for each aircraft. If we can combine these seven tables into one, we will immediately have practical advantages of a rating plan that is both easier to communicate and quicker to calculate. We will also have the advantage that we can see the relative riskiness of craft (according to our rating plan) from one rating table. This would allow an underwriter with experience in the field to challenge the relative risk assigned to each aircraft.[69]

Achieving this is straightforward. We prepare a table that has all the features for each aircraft. Then for each feature, we find the relativity for that table and multiply them together. As an example, the data for the Cessna 172M would be as in Table 8.11. The product of all the relativities shown is 1.37. This compares to relativities of above 3 for various makes of helicopter.

[69] This also helps enormously with model interpretation, as the factors estimated for each of these variables may be distorted by correlation, given the common grain and risk element underlying all of them. Rolled up into a single factor, that correlation is "aggregated away," and the impact of this "area" of the rating plan can be seen and understood more clearly.

Table 8.11. An example of how to calculate the overall relativity for an aircraft.

feature	feature value	relativity
make	CESSNA	1.30
model	172M	1.15
faa_eng_hp_char	B	1.00
faa_eng_thrust_char	Y	1.36
faa_acft_ac_weight	CLASS 1	1.69
faa_acft_type_acft	4	1.00
faa_acft_num_eng	1	0.41
faa_acft_num_seats	4	0.97

8.8. Further Examples

8.8.1. Credit scores

We have mentioned previously that it might be beneficial for certain parts of the GLM to be built separately—even if they are also fitted with a GLM. Credit score is one example.[70] From an operational perspective, there may be some team members or parts of the business (or outside vendors) who are experts in understanding the various credit features, what they mean, and how to use them in a GLM. Scores like this are also very helpful for interactions, allowing pockets of the data to be identified by common score which otherwise would be scattered across disparate combinations of the score components. Such groups would be obfuscated without a score to demonstrate the similarity of their constituent records. There are various possible orders for this process. One example is:

- A GLM is first fitted without the credit features.
- A separate model which uses all the credit features is fitted to the residuals of the first model.
- The result of the credit GLM is a credit score which is then appended to the original data as a new feature.
- A new GLM is fit using the credit score feature.

As previously, the GLM which uses the credit score feature can either fit together with all the other features using the original target or it can be fit on the credit score feature only using the residuals from the first model.

[70] More generally this point applies to any specialized sets of predictors and areas of the data beyond just credit.

We note also that in the above process, where the result of second model is appended to our data as a score and then used in the first GLM, there is no technical need for the type of the second model to be limited to a GLM. With territorial rating, our motivation for using a different model form was that a GLM could not easily "grasp" and allow for the idea of geographic proximity. In this case, our motivation might simply be that it is easier or quicker to use a different type of model or that it results in scores which are more accurate. There are potential practical issues with this approach. Firstly (as opposed to territorial rating where the results of any model type can be stored in a zip code relativity table), the credit score model needs to be applied at the time of quote—and model types more complex than GLMs may not be supported. There is also the issue that the results of more complex model types are not as easily explained as GLMs.

8.8.2. Usage-based insurance

In usage-based auto insurance, the price is often given as a cost per mile. If when developing rating plans, we need to ensure that the cost per mile is constant, we can simply use (the log of) mileage as an offset.

8.8.3. Use of proxy variables

There can be situations where the regulator requires certain variables not to be used (e.g., credit score in some states or gender in Europe). This topic has been addressed in "Implementing Anti-discrimination Policies in Statistical Profiling Models" (Pope and Sydnor (2011)) and in "Discrimination-free Insurance Pricing" (Lindholm et al. (2022)).

Consider the case where one of our features is credit score, and the regulator tells us that our rates must not depend on it in any way. If we exclude credit score from our models and there are other features which are highly correlated with it, they will act as a proxy for it and pick up some of the credit score effect (in the same way that in our example above, age picked up some of the NCB effect). In this case, management might prefer to fit a GLM with both age and credit score and then, in the final rating plan, ignore the coefficient for credit score.

Another case may be that credit score is a permitted variable, but the relativities are fixed by the regulator. Once again, if we fit a model without credit score and use the permitted credit score relativities as an offset, then other correlated features may pick up the difference between the permitted relativities and the true relativities. While this was the desired outcome in our above example with age and NCD, we might not want that here. The alternative is to include the credit score feature in the model and then in the rating plan replace its coefficient with the permitted coefficient. This approach (and others) is discussed further in the above references.

In the above cases, it is important to note that the intercept may need to be re-estimated. Consider the simple example of not being able to use gender in a rating plan and where the multiplicative relativities for gender are 1 for females and 2 for males. Setting the relativity for all customers to 1 without adjusting the intercept will lead to the overall premium across all customers being too low.[71] Whatever adjustment is made needs to be monitored over time, as the proportion of customers of each gender may change over time.

8.9. Practically Speaking

8.9.1. Leakage in two-stage models

Consider a two-stage model where we use the results of a first model in a second model. In the case of our multiplicative model, we will define "residual" as the target variable (let's say this is the number of claims) divided by the predicted number of claims.[72] (The discussion would be identical if we were using preadjusted frequency, see Section 8.5.) If the first model is saturated (for example, a unique identifier for each record was used as a feature so that the prediction from the first model is exactly equal to the actual number of claims for each record), the results from the first model will look something like Table 8.12. The residual being modeled will be one for all observations, and the second stage will fit an intercept-only model.

Table 8.12. num_cl is the number of claims. Predictions from an overfitted first-stage model will render a second-stage model useless.

unqiue_id	fold	num_cl	stage 1 prediction	residual
1	1	5	5	1
2	3	3	3	1
3	2	7	7	1
...				
1000001	9	5	5	1
...				

In general, an overfit first-stage model will reduce the power of the second-stage model.[73] It "uses up" the signal, leaving the second model with less meaningful variation to pick up.

[71] Actuaries often refer to such adjustments as "off-balance" adjustments. See, for example, the interesting discussion in Boor (2022) which notes that off-balance adjustments following credibility and other analyses should not necessarily be the same for all classes.

[72] As discussed above, the weight is also adjusted.

[73] As mentioned elsewhere, it also memorizes the random component of the response, which will not help it better predict outcomes on unseen data.

To avoid this problem, we need, when fitting the second-stage model, to use the "cross-validation predictions." By "cross-validation predictions," we mean the predictions for each record will be from a model trained excluding all records in that fold. For example, the first record above (with unique id = 1) is in fold 1. We use the model which was created on all folds except fold 1 to create a prediction on this record. We do the same for all records. We then use these cross-validation predictions to create the residual which is modeled in the second stage.

In our particular case where the unique identifier itself was used as a feature, the cross-validation model will not be able to make a prediction specific to record 1 and will probably just predict the overall mean. The same will be true for all records. The resulting predictions and residuals are shown in Table 8.13. The two records which have values different from the overall mean (of 5) have residuals different from 1, and so there is a chance that the second-stage model might find something (if there is anything to find).

Table 8.13. cv predictions from an overfitted first-stage model will still leave sensible residuals for a second-stage model to try to fit.

unqiue_id	fold	target	stage 1 cv prediction	residual
1	1	5	5	1
2	3	3	5	0.6
3	2	7	5	1.4
...				
1000001	9	5	5	1
...				

In this section, we used the word "overfit." By "overfit," practitioners sometimes mean that the model has become so overly complicated that while the training performance has improved compared to a simpler model, performance on data not used in training the model has actually deteriorated. In this context, however, we mean any case where a more complex model starts to pick up patterns which are in the training data only and which do not improve performance on unseen data.[74]

8.9.2. Rating groups

A rating plan based on GLM output will consist of a set of tables, one for each feature and interaction. The table for each feature will have one row for each level of the feature. Table 8.14 shows how the table looks for the feature "type of registrant."

[74] That is, data not used for parameter estimation and model structure determination in training.

Table 8.14. Relativities for type of registrant.

type_registrant	relativity
1 (Individual)	0.937
2 (Partnership)	1
3 (Corporation)	2.063
4 (Co-Owned)	1
5 (Government)	1.090
7 (LLC)	1
8 (Non-Citizen Corporation)	1
9 (Non-Citizen Co-Owned)	1

When we make a rating plan change, we will typically want to communicate the whole rating structure, i.e., all the rating tables. Any number of tables of a similar size to Table 8.14 is fairly easy to communicate, whether it be in a spreadsheet or some other manner. However, the table for zip codes contains over 30,000 records when using just the first five characters of zip code. Using the full zip code would lead to a table of millions of rows. To have to communicate this table every time we want to communicate the rates, or changes to them, can be onerous. We can instead break the zip code table into two parts. The first part will map each zip code to a group. The second will match each group to a relativity. For example, Table 8.10 could be restated as Table 8.15. This way, so long as the mapping from zip code to group does not change, we can communicate it just once. The regular rate updates can include just the relativity for area groups.

Table 8.15. Zip code relativities separated into a first mapping between zip codes and area groups and then from area groups to relativity.

zip code	area group
15668	22
21822	62
60077	56
83607	95
85248	40
...	...
...	...
...	...

area group	relativity
1	0.379
2	0.385
3	0.391
...	...
22	0.531
...	...
56	0.917
...	...

(Note that the difference of 1.0162 between each group was derived as follows: in our case, the minimum zip code relativity is 0.379 and the maximum is 1.851. The (multiplicative) difference is 4.884. If we are prepared to have 100 groups, then the (multiplicative) difference between each group is 1.0162, that is, $1.0162^{99} = 4.908$).[75]

The above separation of area rating into two separate tables serves another useful purpose. A separate team or project can work on allocating zip codes to high- and low-area groups. The first-stage model can then be fit with area group as a feature.

The above discussion applies equally to aircraft groups, where the underwriters may (or may not) find it easier to discuss the Cessna 172M being in group 5 and the Bell 47G helicopter being in group 95 rather than talking about relativities of 1.37 and 3.13. In general, splitting out the tables in this way gives transparency and interpretability. We can easily show the composition of the final rate, and business experts can more easily challenge it.[76]

[75] While this is one sensible way to create the relativities, it is not the only such way.
[76] ...which is a good thing.

9. Learning From Black-Box Models

Black-box models are machine learning models whose predictions are hard or impossible to explain. Our penalized GLM is a machine learning model which is easy to explain. Its predictions are the product of numbers which depend in a simple way on our original features. Our benchmark model, a gradient boosting machine, is close to impossible to explain. It is made from 1,000 decision trees, each tree having many decisions, each of which contributes in some small way to the final prediction. However, it performs better than our GLM. On the FAA-NTSB case study in Section 5.9, the performance of our GLM was about 10% lower than the performance of the gradient boosting machine (GBM). Our preference (and in many cases the regulator's) is for rating plans whose prices are easy to explain—such as those derived from GLMs and generalized additive models (GAMs). We also want to compare the GLM to the GBM and to learn from it if we can. In which ways are the models similar or different? Where we have taken a specific modeling approach—for example, regarding nonlinear effects— how do our predictions compare to the GBM? We do not want to sacrifice performance if we can avoid doing so. Can we find out how the GBM is outperforming our GLM and incorporate this into our model? We will look at three areas:

- Variable selection and importance
- Nonlinear effects
- Interactions

As we have made a significant effort to deal sensibly with the first two items, but have not dealt with the third at all, we expect that if we find ideas to increase the performance of our GLM, it will be with interactions. Nevertheless, we will check all three points.

We will not review the differences in the approach to HCCVs between the two models—because they are essentially the same. In fact, were it not for the fact that the GBM implementation we used has a built-in method for dealing with HCCVs, the approach would have been exactly the same, i.e., we would have created a target encoding independently of the type of model we were using and then used that encoding in both model types. We did however refit our GLM and the benchmark model without

the make-model HCCV discussed earlier—and the differences between the models were more or less unchanged.

The methods we will use in this chapter do not depend on the type of model we used as a benchmark. They would work equally well with random forests, a neural network, or any other model. Such methods are known as "model-agnostic." (Methods which depend on the model type are called "model-specific" methods.)

Most of the ideas in this chapter are explained in detail in *Interpretable Machine Learning; A Guide for Making Black Box Models Explainable* (Molnar (2020)). In our work, we use only a small subset of the techniques mentioned there that are particularly useful to us as we cover the points above. For each area we discuss in this chapter, there are many other approaches. Our aim is to show generally the kinds of questions that can be asked, rather than to exhaustively provide answers (even if that were possible).

Before looking at the above three items, we will discuss one risk of learning from complex black-box models.

9.1. Model Variance Revisited

We saw in Figure 4.10 that, as penalization reduces, our GLM can get more and more complex and as a result might start to suffer from model variance. That is, it might adapt itself to pick up patterns in the training data which are not in the validation folds or in the test data. While it is always tempting to choose the model with the very best cross-validation performance, if this model has a large gap between training and cross-validation performance, then there is a risk that our pricing will include claims patterns for certain segments of our customers that are just random noise and will not be present in future data. In insurance settings, this is especially risky as it might lead to an increase in the proportion of our future customers in those segments and an overall poor performance.

The GBM benchmark that we have used is essentially a phenomenally complicated GLM. It allows numeric features to be built from an almost unlimited number of step functions, and it allows an almost unlimited number of interactions. In fact, if we engineered a very large number of step functions and interaction features, we could express our GBM in the form of GLM. Given its complexity, we should not be surprised if it does suffer from large variance; the equivalent GLM would too. As each tree gets added, the GBM gets more complex, so one way to see how model variance increases with complexity is to plot the train and cross-validation performance as the number of trees in the GBM increases. Figure 9.1 shows that there is a significant gap between train and cross-validation performance as the number of trees increases.

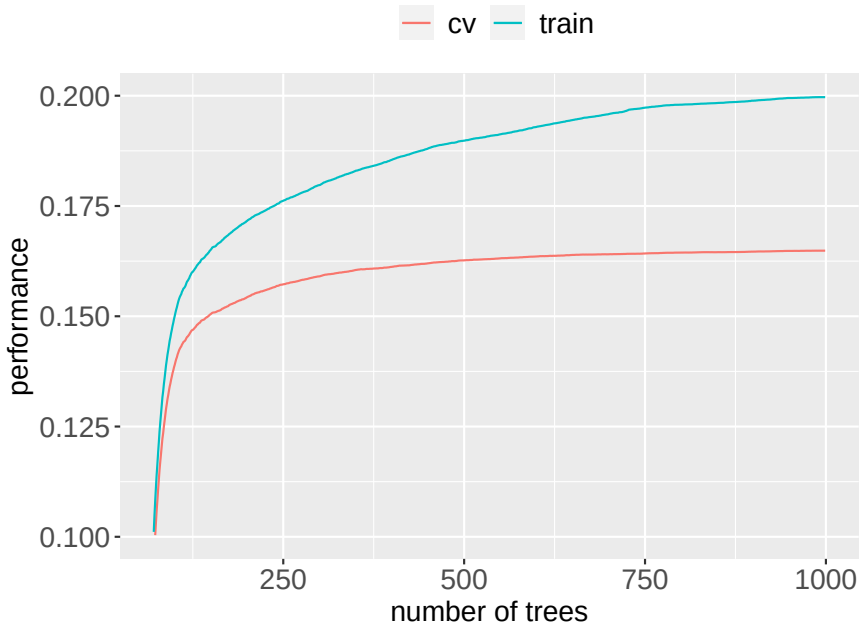

Figure 9.1. The gap between train and cross-validation performance for the benchmark GBM. Even at performance levels of around 0.15 (achieved by our GLM), and certainly at higher levels, the gap is wide, suggesting that the GBM is picking up patterns in the training data which are not in the cross-validation partitions—it suffers from high variance.

Given the above finding, if we attempt to incorporate patterns that the GBM has learned into our GLM we will need to monitor the variance of the GLM model and take appropriate action. These could include excluding patterns that increase variance or accepting the increased variance and putting processes in place to compare the mix of business written after implementation of the new rates to the mix of business in the training data.[77]

9.2. Variable Selection and Importance

We would like to know if we have included the right features in our GLM and haven't overlooked any important features from the benchmark model. This is not a straightforward question to answer. While the LASSO (for example) will completely exclude from the GLM the features that it does not select, tree-based and certain other non-GLM machine learning models often include all features. It is just that some of the features will play only a very small role in the value of the final predictions. (The reasons why some machine learning models tend to include all features are discussed briefly in Section 9.6.)

[77] This is something which should be done in any case.

Given the above, rather than asking which features are in our GBM, we ask which features are important to our GBM. We have previously seen that for a GLM, the magnitude of the coefficients for the standardized features can be considered as feature importance. This method is not available for GBMs—they do not have coefficients. Instead, the following simplified example shows how we can understand the importance of features in any model.

In our example, there are three insureds with ages 20, 30, and 40. Frequency is $age/100$, and there is no randomness. We also have the height of each insured but frequency does not depend on height. Our trained model has correctly worked out that $freq = age/100$. If we knew what resulted from our trained model, we would know that age was a very important feature and height was not at all important. If we don't know how our trained model works, we only have access to the predictions that it makes. So we first ask our model to create predictions on a dataset held out from its training—the results are shown in Table 9.1. The predictions are—unsurprisingly—perfect. For simplicity, our performance metric will be mean absolute error, which is clearly zero.

Table 9.1. The start of our simple example to illustrate a type of variable importance calculation. We have the ages and heights of three insureds. Frequency is age/100. There is no randomness, and our trained model has learned the true relationship—though we do not know this or the form of the model that it has learned.

age	height	true frequency	num_cl	predictions	absolute error
20	1.4	0.2	0.2	0.2	0
30	1.5	0.3	0.3	0.3	0
40	1.6	0.4	0.4	0.4	0

We will now take our original data but randomly permute the age column and keep all other features—in this case, height—the same. The true frequency remains the same. We ask our model to provide predictions and, of course, this time the predictions are all wrong because the age data has been made rather useless by its being randomly permuted. The results are shown in Table 9.2, and the mean absolute error (MAE) is now 0.133. The increase to MAE, as a result of making the age feature useless, is 0.133.

Table 9.2. Stage 2 of our simple example. We permute the age column, ask our model for a new set of predictions, and recalculate model performance.

age	height	true frequency	num_cl	predictions	absolute error
40	1.4	0.2	0.2	0.4	0.2
20	1.5	0.3	0.3	0.2	0.1
30	1.6	0.4	0.4	0.3	0.1

Next, we take our original data and randomly permute the height column and keep all other features—in this case, age—the same. The true frequency remains the same. We ask our model to provide predictions. This time the predictions are unchanged because our model does not depend on height (though we do not know this up front). The results are shown in Table 9.3, and the MAE is now zero. The increase to MAE, as a result of making the height feature useless, is zero.

Table 9.3. Stage 3 of our simple example. We permute the height column, ask our model for a new set of predictions, and recalculate model performance.

age	height	true frequency	num_cl	predictions	absolute error
20	1.6	0.2	0.2	0.2	0
30	1.4	0.3	0.3	0.3	0
40	1.5	0.4	0.4	0.4	0

Finally, we have for each feature the amount of performance lost by making that feature useless by randomly permuting its column in our data. The worse the loss of performance from permuting a feature, the more important we say it is in our model. This type of feature importance is known as "permutation-based feature importance." For convenience we divide each value by the maximum value so that the "most important" feature has a feature importance of 100 and all other values are relative to that amount.[78]

[78] We have introduced MAE here for simplicity. In any rating exercise we would expect the calculations done above to use the same performance metric being used in the rest of the rating exercise. For example, for feature importance calculations for our FAA-NTSB study we used pseudo-R^2. In the final presentation of the feature importances one might then choose not to rebase to 100% as the raw pseudo-R^2 figures will show how the performance we are familiar with reduces as features are rendered useless.

Table 9.4. The final stage of our simple example. The importance of each feature is the amount by which performance got worse when that feature was made useless by permuting it. For convenience we divide each value by the maximum value so that the "most important" feature has a feature importance of 100 and all other values are relative to that amount.

feature	increase to MAE	permutation-based feature importance
age	0.133	100
height	0	0

The advantages of permutation-based feature importance are its ease of computation for any model and the fact that it requires only the ability to generate predictions for a dataset. Hence the approach is model-agnostic, i.e., it can be applied to any model. In addition, we have defined an "important" feature as one which helps the model to predict accurately. When applying this approach, we can use data which was not used to train the model, so an important feature is one which helps our model to predict accurately over future data—which is what we really care about. (Alternative approaches consider a feature important if changing it leads to big changes in predictions or if it is important in some way in the trained model itself—for example, did the model use it to make many splits in the tree?)

A disadvantage of this method is that, if our features are correlated, permuting only one column at a time may create observations which are impossible. For example, in our FAA data, engine type can be turboprop or jet. The power of turboprop is measured in horsepower and is included in faa_eng_hp. The thrust of jet engines is measured in lb of thrust and included in faa_eng_thrust. When we permute the engine type column only, we will end up with jet engine observations with horsepower measurements. We have not just made the contribution from the engine type feature useless, we have made the whole observation meaningless.[79]

We need to remember that we are finding the importance of features in our final trained model, not the importance of features left out of our final model. Imagine that an exceedingly important feature has been excluded. The above approach will give it an importance of zero. In our case, this is not an issue: we are using feature importance to compare the differences in importance between two different models. We are asking if there is an important feature in the GBM which has been left out of the GLM, not if

[79] A separate but related point is that it is often helpful to permute variables in groups. Groups are selected either due to correlation or because they cluster together logically. This formulation can be a useful addition to the basic permutation importances, since seeing the importance of variable groups (e.g., driver, vehicle, and policy level features in personal passenger auto insurance) provides insights not otherwise available.

there is a feature in the original data which has been left out of both models. We are also not determining the importance that the features in our final trained model could have had in some other model.

Consider two very highly correlated features, both in our final trained model, and our model generally uses one of them and not the other. The one which is used most will have a higher permutation-based feature importance, despite the fact that in another trained model it could have been left out completely in favor of the other.

Finally, we note a point which will become useful later. When we permute a column, we don't only make it useless. We make any interactions which rely on it useless. Hence, permutation-based feature importance for a given feature measures its importance both as a main effect and in any interactions in which it is involved.

We calculated permutation-based feature importance for our GLM and the benchmark GBM. The six least important features for our GLM are shown in Table 9.5. For comparison, we show at the top the most important feature. We can see that the GBM agrees that these features are relatively unimportant—all of them have an importance of less than 1% compared to region.

Table 9.5. The six least important features for our GLM. Both feature importance values and ranks are similar to those of the GBM.

	feature importance value		feature importance rank	
feature	GLM	GBM	GLM	GBM
region	100.0	100.0	1.0	1.0
street2_ind	0.07	−0.02	16.0	20.0
co_ownership	−0.001	0.08	17.0	18.5
co_owners_num	−0.01	0.23	18.0	16.0
faa_acft_type_acft	−0.02	0.58	19.0	13.0
faa_acft_ac_weight	−0.06	0.08	20.0	18.5
faa_acft_ac_cat	−0.12	−0.05	21.0	21.0

We note that none of the feature importances are exactly zero for the GLM. This is surprising given that it was penalized with the LASSO which carries out implicit feature selection. This is partly due to the data and GLMs we used to calculate feature importance, and this is discussed further in Section 9.5.1. Note also that some feature importances are negative. This suggests that we should consider refitting our GLM without these features. In our case, given that we have used a solid performance based approach to including features in our GLM, we would be inclined to believe that these negative values are likely related to randomness in our data and the disadvantage of permutation-based feature importance mentioned above (that it can create impossible observations).

Table 9.6 shows the 15 most important features for our GLM. Both feature importance values and ranks are similar to those of the GBM. However, there are some exceptions. In particular, the number of aircraft registered to the owner is much more important in the GBM than it is in the GLM; it is the third most important in the GBM, but only the 11th or 12th most important in the GLM.

Table 9.6. The table shows the most important features for our GLM and GBM. Mostly they are similar; however, there are some exceptions. In particular, the number of craft registered to an owner (nu_registered) is much more important in the GBM than it is in the GLM.

	feature importance value		feature importance rank	
feature	GLM	GBM	GLM	GBM
region	100.0	100.0	1.0	1.0
veh_age	44.9	47.6	2.0	2.0
type_registrant	2.7	3.1	3.0	4.0
operation	2.6	2.4	4.0	6.0
faa_acft_num_seats	2.0	2.1	5.0	7.0
airworthiness	1.9	1.4	6.0	8.0
faa_acft_speed	1.5	2.8	7.0	5.0
faa_acft_num_eng	1.1	0.7	8.0	11.5
faa_eng_thrust_char	0.7	0.9	9.0	10.0
faa_acft_type_eng	0.6	0.7	10.0	11.5
nu_registered	0.5	4.0	11.5	3.0
faa_acft_build_cert_ind	0.5	0.3	11.5	15.0
faa_eng_type	0.4	1.2	13.0	9.0
kit_indyn	0.3	0.4	14.5	14.0
faa_eng_hp_char	0.3	0.1	14.5	18.0

Based on the above discussion, we conclude that we have not left out any features from the GLM that were important in the GBM. However, we have missed something with the feature for the number of aircraft registered to the owner. There are two possibilities here; we have incorrectly fitted the main effect, or there is an interaction between this feature and one or more of the other features which we need to add. We will cover these two points in the following sections.

Before moving on, we note that in our case, we only have a relatively small number

of features. In such cases, using feature importance to check on the feature selection does not necessarily add much value. In cases where there are large numbers of features (hundreds or thousands) this approach becomes increasingly useful as a reasonability check and in providing candidate features to add back to the GLM.

9.3. Nonlinear Effects

How can we compare the treatment of numeric features that affect our target variable in a nonlinear way? It is easy to plot the average prediction over our training data for each value of the feature. We can do this for both the GLM and GBM and compare the graphs. Figure 9.2 shows the average predicted frequencies for the number of aircraft registered to the owner. We have capped the values shown on the x-axis at 30—the actual values above this are sparse and extend beyond 3,000. The predictions from the two models are mostly similar (especially for values where the proportion of exposure is high), though predictions from the GBM are lower for values of nu_registered between 11 and 15. While the GBM may indeed be correct (we will discuss this further below), given the limited amount of exposure, it is unlikely to be the reason why this feature has a much larger importance in the GBM.

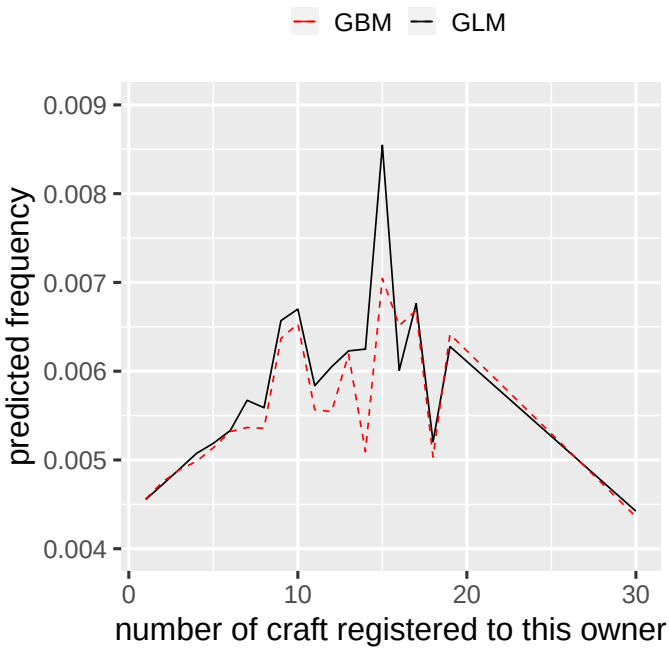

Figure 9.2. A comparison of the average predictions from the GLM and GBM. Values shown on the x-axis are capped at 30—the last point including exposure for all craft with nu_registered >= 30. Predictions from the GBM are lower for values of nu_registered between 11 and 15, but it would be surprising if this difference, applying to only a limited amount of exposure, explains why this feature should have a much larger importance in the GBM.

The above approach does not isolate the impact that the feature has on the predictions. For example, the average prediction where the feature is 30 or more depends not only on the feature, but also on the values of all other features for owners who have 30 or more craft registered under their name. We therefore consider a different way of visualizing how predictions depend on a feature.

For GLMs, especially where there are no interactions, we can very easily isolate the impact of a given feature on the predictions—it is expressed by the value of the coefficients. The GLM we are using in this chapter uses step functions for nonlinear effects, and the linear predictor for nu_registered is calculated as follows:

$$\begin{aligned}\text{linear predictor} = &-0.0001407 \times \text{nu\_registered} \\ &+ 0.02936 \text{ if nu\_registered} \geq 2 \\ &+ 0.07342 \text{ if nu\_registered} \geq 7 \\ &- 0.15822 \text{ if nu\_registered} \geq 16 \\ &- 0.17117 \text{ if nu\_registered} \geq 35\end{aligned}$$

Taking the exponent gives us the relativities. Figure 9.3 shows the relativities, rebased to be one for nu_registered = 1. We can see that the overall shape is made up of four different levels arising from the four step functions, plus a small negative slope (which would be more apparent if we extended the x-axis to higher values). The impact on predicted frequencies is easy to see and easy to challenge, and indeed we would certainly challenge and likely adjust such relativities before their use in a rating plan.

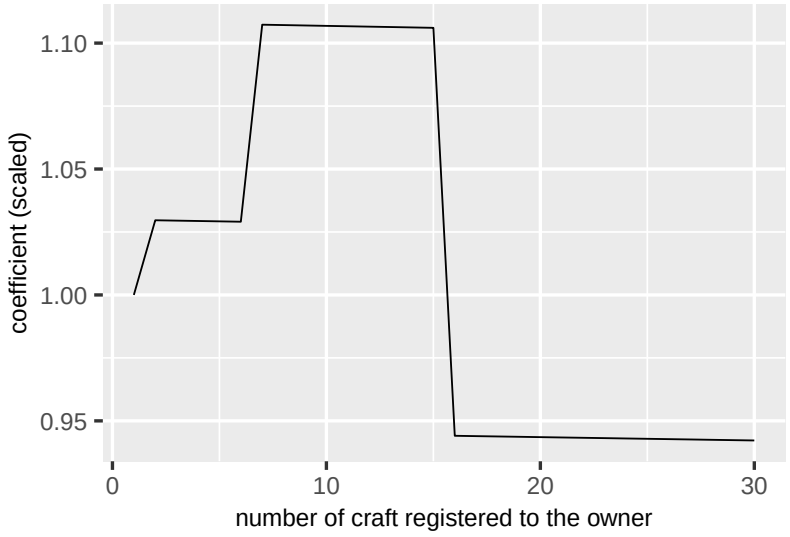

Figure 9.3. The relativities for nu_registered in our GLM.

The GBM does not have coefficients, so we cannot provide a graph for it on the same basis. We therefore consider an alternative approach, first using our GLM as an example and then applying it to our GBM. We know from the above graph that (compared to nu_registered = 1) the relativity for nu_registered = 10 is 1.107. If we did not know the GLM coefficients, we could still find this out using only our ability to calculate predictions. We first set nu_registered to 1 for every policy in our data and calculate the prediction. The first few lines of our data are in Table 9.7.

Table 9.7. Predictions from a GLM model having set nu_registered to 1 for all policies.

nu_registered	faa_acft_num_seats	craft age	other features	pred
1	2	1	...	0.00436
1	9	3	...	0.00109
1	2	18	...	0.00048
1	4	0	...	0.00482
1	6	27	...	0.00032
			Total	0.01107

Next, we set the value for nu_registered to 10 for every policy and again calculate the predictions. The revised predictions are shown in Table 9.8.

Table 9.8. Predictions from a GLM model having set nu_registered to 10 for all policies.

nu_registered	faa_acft_num_seats	craft age	other features	pred
10	2	1	...	0.00482
10	9	3	...	0.00120
10	2	18	...	0.00053
10	4	0	...	0.00533
10	6	27	...	0.00036
			Total	0.01225

Finally, Table 9.9 summarizes our results, showing the predictions for the two values of the feature and the ratio between them. Given that the GLM has no interactions and the only thing that changes is the value of nu_registered, the ratio of the predictions on each line, and in total, must be the ratio of the relativities for nu_registered = 1 and nu_registered = 10.

Table 9.9. We divide the predictions from setting nu_registered to 10 by those from setting nu_registered to 1. This must be the ratio of the relativities for those two values, since nothing else has changed.

pred_1	pred_10	ratio
0.00436	0.00482	1.107
0.00109	0.00120	1.107
0.00048	0.00053	1.107
0.00482	0.00533	1.107
0.00032	0.00036	1.107
Ratio of totals		1.107

We repeated this exercise for every value of nu_registered, and indeed the resulting relativities and graph are identical to that produced from the coefficients above. Had there been interactions with nu_registered in the GLM, then the ratio for each policy would depend not only nu_registered, but also on whether or not it interacted with any level of any other feature for that policy. We could still take the ratio of the totals, however, and this would tell us, on average across all policies used in the calculation, how the predicted frequency changes when nu_registered is increased from 1 to 10.

We now do exactly the same for our GBM. Unsurprisingly, the ratio on each line is different; however, we can still find the ratio of the mean predictions. The result is 1.104. While this is similar to the GLM, we will soon see that for values of nu_registered other than 10, the values are quite different.

Table 9.10. Predictions from the GBM, after first setting nu_registered to 1 for all policies and then to 10. The ratio of the average predictions is 1.104—which is similar to that of the GLM.

unique_id_line	pred_1	pred_10	ratio
1	0.00323	0.00389	1.206
2	0.00132	0.00228	1.729
3	0.00053	0.00075	1.412
4	0.00330	0.00387	1.172
5	0.00040	0.00051	1.285
...	...	...	...
Average prediction	0.005336	0.005892	1.104

Once again, we can compute the ratio for each value of nu_registered. Because we look at how our model's predictions depend on just one or a small subset of the features, the approach is called "partial dependence" (Friedman (2001), Section 8.2). The plot of

the partial dependence values for the full range of our feature is known as a partial dependence plot (commonly known by the acronym "PDP"). Figure 9.4 compares the partial dependence plots for the GLM and the GBM. We can see that the two plots are quite different. The biggest difference is in the 11–15 range (although they also differ elsewhere), and we know from our previous discussion that in this range the GBM predictions are lower than the GLM predictions. A review of the step functions that were fed into the LASSO shows that due to the limited exposure in this range, there was no step in this range (there were step functions at nu_registered 2, 3, 5, 7, 10, 16, 35, and 283). While we don't know at this stage whether or not the pattern the GBM has spotted will generalize to test data and future policies, we can certainly rerun the GLM with some extra detail in the step functions.[80]

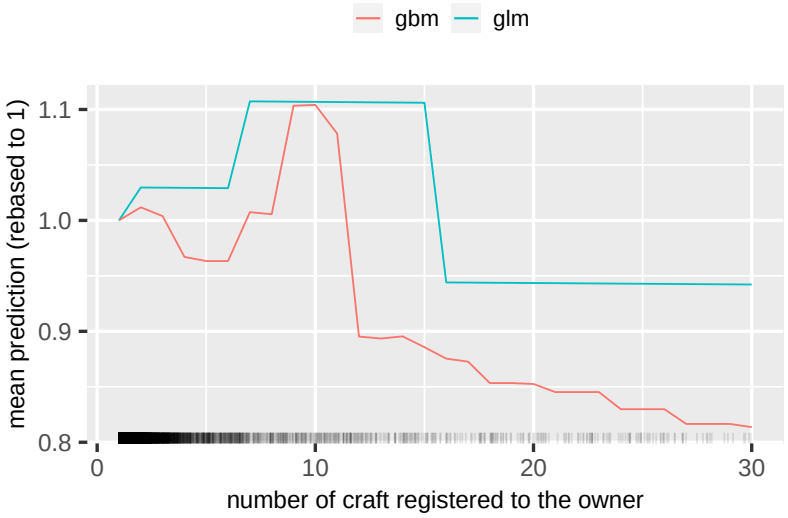

Figure 9.4. A comparison of the partial dependence plots for the GLM and GBM.

In this section, we have seen how we can use our benchmark black-box model to check and if necessary adjust our GLM approach to numeric features with nonlinear effects. Before we move on, given that we have introduced partial dependence, we mention some caveats in its use. Firstly, as with our permutation-based approach to feature importance, simply changing all policies to take a given value of a certain feature can create impossible observations. More importantly, partial dependence is a global method—it averages the dependence of predictions on a feature over all observations in our data. The actual impact will almost certainly depend on other features (due to the myriad interactions present in most black-box models).

[80] This is an exercise left to the diligent reader.

For example, Table 9.10 shows a wide range of ratios of prediction values when nu_registered is changed from 1 to 10 (ranging from 1.172 to 1.729). Figure 9.5 shows a histogram of the ratios, and the red dashed line shows the ratio of the mean predictions, which is what we used in our partial dependence plot. The histogram makes very clear that ratios at the level of each observation span a wide range. Friedman (2001) caveated the usefulness of the approach used here in exactly this situation.

The red dashed vertical line in the histogram is drawn at 1.104. As discussed above, this represents the ratio of mean predictions between nu_registered being set to 10 and 1. Our method of calculating this ratio effectively weights each observation by its predicted frequency. The fact that the red dashed line is so far to the left of the distribution is because the predictions for which the ratio is lower than 1.104 are in general far higher than for cases where the ratio is high. This again reminds us that partial dependence is a global method and the impact of a feature on various segments of the portfolio may be very different from that in the partial dependence plot.

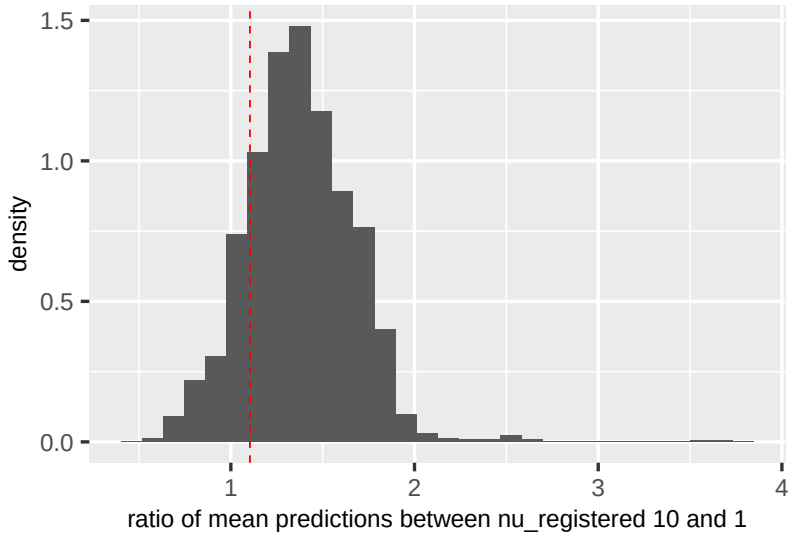

Figure 9.5. A histogram of the ratios between catboost predictions where nu_registered is set to 10 compared to being set to 1. The red dashed line is drawn at 1.104, being the ratio of the mean predictions across the whole dataset. The histogram demonstrates that there is a very wide range to the ratios—quoting the global value of 1.104 can be very misleading when thinking about any one individual policy. The red dashed line is far to the left because policies with low ratios have much higher predictions and so have a bigger weight in the overall mean.

Molnar (2020) presents various alternatives to partial dependence plots to deal with the challenges raised above, and we refer the interested reader to that excellent work. We, however, move on now to consider interactions.

9.4. Interactions

There are many ways to identify and fit interactions when fitting GLMs, and our work does not aim to cover this topic. However, we do wish to know whether the performance differential between the GLM and GBM is due to interactions being important in the GBM and being completely missed out from the GLM. We will briefly cover two points:

- Are there any interactions in our GBM which are particularly important?
- If we feed a basic approximation of the key interactions into our GLM as features, does its performance improve?

Given that we have introduced permutation-based feature importance above, we now introduce a somewhat simplistic method to use it to rank pairs of features to check for interactions. Since permutation-based feature importance is a model-agnostic method, so is this approach to checking models for interactions. Consider two features, A and B, which do not interact in our model in any way—neither with each other, nor with any other feature. When they are permuted, they reduce performance solely to the extent that predictions depended on them. Say that performance reduced by 1% for A and 1% for B. If they are now both permuted at the same time, performance will reduce by 2%. The sum of the loss of performance from their separate permutations minus loss of performance from permuting both features at the same time is zero.

Consider again two features, A and B, which are separately important but now also interact with each other. When we permute any one of them, we destroy not only its performance but also the performance due to the interaction. Say their own performance is 1% each and the performance due to the interaction is also 1%. In this case, when they are separately permuted, the loss of performance is 2% each time, and when they are jointly permuted, the loss of performance is 3%. Now the sum of loss of performance from their separate permutations, minus loss of performance from permuting both features at the same time, is 1%. (This argument also holds where both features interact with the same third feature but not with each other.)

Based on the above argument, if pl_A and pl_B are performance loss from separate permutations, and pl_{AB} is performance loss from joint permutation, then the extent to which

$$pl_A + pl_B - pl_{AB}$$

is positive indicates either that A and B may interact with each other or interact with a common third feature. Therefore we can rank all pairs of features by this measure and check the top few visually for interaction.

Programmatically, it is simple to permute each feature separately to find and then to also permute all combinations or two features and to find the difference as described above. Some of the calculations for interactions with aircraft age are shown in Table 9.11.

Table 9.11. There are many possible pairs of features that may interact in our model. This table shows the calculations for three of them. The method used suggests interactions of aircraft age with region and nu_registered.

feature_A	feature_B	pl_{AB}	pl_A	pl_B	$pl_A + pl_B - pl_{AB}$
region	aircraft_age	0.291	0.253	0.135	0.096
nu_registered	aircraft_age	0.115	0.016	0.135	0.037
faa_eng_hp_char	aircraft_age	0.123	0.002	0.135	0.014

The results over all pairs of features are shown in Figure 9.6.

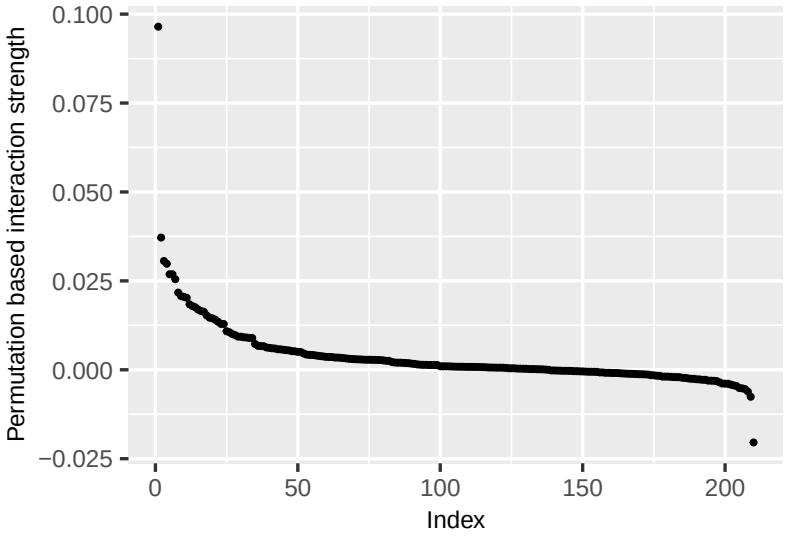

Figure 9.6. This figure shows all pairs of features in our data, sorted according to the size of our permutation-based interaction strength measure. The top proposed interaction is between region and aircraft age.

In a typical model-building exercise, we would now inspect, graphically or otherwise, the top few candidate interactions. Figure 9.7 shows an example of a graphic that might be used to inspect an interaction. Aircraft ages have been grouped into reasonably sized buckets, and region (being the location of the registered owner) has been grouped into two groups: the US and elsewhere. The y-axis shows the percentage error in predictions for each bucket (based on actual claims/predicted claims).

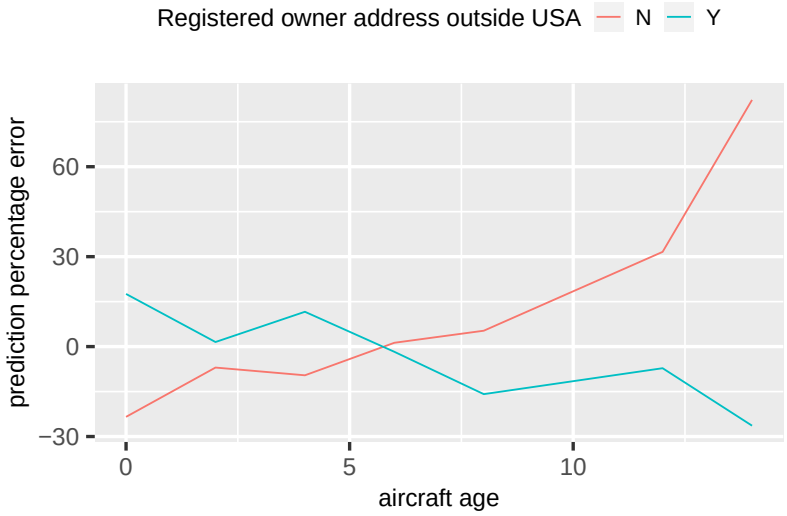

Figure 9.7. This figure is an example of how we might inspect our data to see if there is an interaction between region and aircraft age. Aircraft ages have been grouped into reasonably sized buckets, and region (being the location of the registered owner) has been grouped into two groups: US and elsewhere. The y-axis shows the percentage error in predictions for each bucket (based on actual claims/predicted claims). It can be seen that there might be benefit in allowing the interaction into our GLM so that at lower aircraft ages, predictions are higher for regions outside the US and lower for regions in the US—and the opposite at higher ages.

We have now seen how we can get a list of candidate interactions from our black-box model and shown briefly how we might inspect them further. Each candidate interaction needs separate checking and fitting with the GLM—depending on how this is done, it can be a fair amount of work. It would be useful to have a quick way to check in advance of all this work whether inclusion of interactions will improve the performance of our GLM. A fairly simple way to do this is based on decision trees. A single decision tree divides our data into separate pieces using a limited number of interactions. Each observation that we ask our decision tree to predict on will be given a leaf number, and the prediction will be the average of all the training data that was sent to that leaf. We will not be interested in the decision tree predictions, but the leaf numbers—each of which represents an interaction—are useful. We therefore start by dividing our black-box predictions by the GLM predictions. Any differences will be mainly due to the interactions in the GBM.[81] We then model these differences using a simple decision tree. An example of an interaction represented by one leaf in the resulting tree is shown below:

[81] A tree model estimates all effects as interactions, including effects which aren't interactions. Any main effect the GLM missed will be picked up by the tree, and the tree's leaf node then may, or may not, represent a real interaction effect. In our case, we have checked that the GLM has not missed out on any main effects, and we therefore expect leaf nodes to represent interactions.

```
node number: 24
root
nu_registered >= 22
aircraft age <= 3
region = X (outside the US)
```

For this subset of observations, the black-box predictions are 65% lower than those of the GLM. However, we are more interested in the interaction itself—we want to feed this into our GLM. We can create a new feature in our training data which labels each observation with the leaf number that it falls into. We use this new feature as the only feature in a penalized GLM (with our original GLM predictions as an offset). Using cross-validation as normal, we find that our GLM pseudo-R^2 improves to 0.156. While this still falls 6% short of our gradient boosting benchmark (pseudo-R^2 of 0.165), it suggests that with further effort, we should be able to improve our GLM performance using correctly selected and fitted interactions.

This brings us to the end of our discussion on how we can check our work using black-box models and how we can bring insights from them back into our GLM framework.

9.5. Practically Speaking

9.5.1. Feature importance—train or test data

As discussed above, as our models get more complex, there is a risk that patterns are found in the training data which will not be present in future data. If a feature is important on training data, it does not mean that it will be important for performance on future data. To know which features are important in the final model which is trained over all of our training data, we should base importance on predictions and performance over our test data. However, during modeling, we do not want to look at the test data. Therefore, the permutation-based feature importance reported in this chapter were based on the models created during our cross-validation process. For each of these, we calculated feature importance of all features on the held out fold. If we have 10 folds, then we have 10 cross-validation models, and for each feature we end up with 10 estimates of its importance. The feature importances that we reported above are, for each feature, the average of the importances across all folds. The importances for the LASSO are not exactly zero for any feature because there is no feature which was penalized out of all of the cross-validation models.[82]

[82] When using the LASSO it is certainly possible that some features will be penalized out of all the cross-validation models, and then their feature importances would indeed be exactly zero when calculated in the manner discussed here.

9.6. Technical Note

9.6.1. Reasons that certain model types include all features

The reasons why some machine learning models tend to include all features are specific to the various model types. In random forests, each single tree can be—and often is—overfit. This ensures that any one tree has low bias. The averaging over many trees reduces the variance and can potentially result in a good ensemble. Since each tree is overfitted, each tree is likely to contain most of the features that it was allowed to use. GBMs are made up of hundreds or even thousands of trees, each one taking a small step in the direction of the optimal solution given all the trees that came before it. These small steps will often pick up some of the noise in the dataset, and hence, for any feature, there are likely to be at least some trees that contain that feature. Neural networks learn their weights by iteratively increasing the magnitude of weights to connections that improve performance and reducing the magnitude of weights to those that reduce performance. Features which are not important will tend to end up with low weights rather than zero weights.

10. Challenges and Considerations in Ratemaking Models

In this monograph, we have focused on certain technical challenges that occur during the training of GLMs. Throughout the process and after, we need to be aware of the limitations of our data and the impact this may have on the performance of our models. In this chapter, we look at some of the issues that a practitioner should consider.

10.1. Understanding the Data

Before working with a dataset, we need to understand it. We need to be clear on the meaning of each field and the range of values we should expect. In order to do this it is a good idea to speak to the team that created the dataset and/or to people who have previously carried out analysis on the data. Likewise it is useful to speak with underwriters or others who understand the line of business.

In our work, we made a basic error which could have been avoided with additional conversations prior to carrying out our analysis. During data preparation, we noticed that the age of manufacture was missing in about 20% of cases, which meant that for those cases we could not calculate the age of the aircraft. We could have set aircraft age to missing for those cases and then dealt with missing values in one of a number of ways. However, on the assumption that the certificate (for which the data contains the feature "certificate_issue_date") is issued shortly after manufacture, a "shortcut" seemed to be to use the certificate issue year instead of year of manufacture. We later realized that registration certificates (as opposed to airworthiness certificates) are issued at various stages during an aircraft's life, in particular upon change of ownership or use. This is apparent from the data as well but is an error that would certainly have been avoided with discussions prior to analysis.[83]

Other areas where discussions would have benefited our analysis are in the treatment of drones and in the treatment of aircraft for which we assumed that the registered users

[83] We have not redone the analysis including the corrected aircraft age and years since aircraft certificate issue. We leave that as an exercise to the interested reader.

have addresses outside of the US.[84] It is possible that some of the strong interactions seen in the previous chapter are due more to aspects of our data preparation than true differences in the probability of accidents.

10.2. Technical Errors

There are errors that we practitioners should not make during modeling, but we do. At best, they can waste time by requiring us to rework an analysis; at worst, they can negatively impact our results and cause incorrect rating plans to be implemented.

Examples include:

- Inconsistent coding of rating variables between modeling train and test datasets or, worse, between analysis data and operational data. A simple example of where this can go wrong is where variables which are strings are coded to categorical variables, e.g., Cessna is coded as 1 and Piper is coded as 2 in training data, but in test data the coding is reversed. In certain software, this coding is mostly invisible to the user and needs to be explicitly checked.
- Coding categorical features as numbers that are then treated in an ordinal manner. In our FAA-NTSB work, we intended to treat the number of engines as a categorical feature, but early on we were treating it as a number. An easy way to ensure that this does not happen is to store categorical features using letters (for example, one engine is coded as "A").
- Including segments of data which are not relevant and whose experience might adversely impact areas of the model where performance matters. For example, drone insurance may be sold under a separate product and rating plan. Including a significant amount of drone exposure (which has no claims) in our data is unlikely to help performance on other general aviation business.

These and a multitude of other technical errors are easy to make. Peer reviews during modeling and after implementation of the rating plan are required to avoid, or at least reduce, such errors.

10.3. Data Does Not Meet Model Assumptions

Insurance data often contains inherent dependencies that challenge the assumption of independent and identically distributed (iid) observations, which is a foundational premise for most supervised learning techniques, including GLMs and their variants. For example, someone who has previously filed a claim is more likely to file another in the future, which is why insurers often apply surcharges for recent claim experience. Similarly, in-

[84] Our assumption that the US region being missing meant that the registered user has an address outside of the US was wrong.

suring two neighboring townhouses introduces a dependency because a fire in one could easily spread to the other, a scenario unlikely if the houses were miles apart.

These examples highlight that our data are often not independent and, while we can still build effective models despite violating the iid assumption, it's crucial to verify and account for this lack of independence. In such cases care must be taken in setting up cross-validation folds to avoid data leakage. For instance, in personal auto insurance, all vehicles under the same policy should be placed in the same fold, and for weather-driven perils, each year could serve as a separate fold to prevent storms from leaking across them.

Provided that this is done, we can then benefit from the power of cross-validation to help measure the uncertainty of our model's predictions, even in the presence of non-independent data. By dividing the data into folds and training and testing in various combinations, cross-validation can provide insights into how the lack of independence impacts our model's performance. While cross-validation does not alter the final model, it can inform decisions about model adjustments and offer a more accurate assessment of how dependencies in the data might affect the model's predictions.

10.4. Stability Over Time

10.4.1. Coefficients can change over time

Model drift (or concept drift) refers to the situation when a model's performance gets worse over time because the relationship between the predictors and the response variable changes over time. For example, the risk associated with young drivers might increase due to rising distractions, or underwriting discounts might evolve into surcharges if they are gamed by insureds. Some changes may be quite sudden and may also reverse at some point. For example, for private auto portfolios which use occupation as a rating factor, the relativities for different occupations may have changed during COVID-19.

Generalized linear models (GLMs) and generalized additive models (GAMs) can be vulnerable to time inconsistencies because they tend to average out such inconsistencies in the final coefficients, potentially masking important temporal trends. Therefore checking and accounting for the time consistency of parameters is a critical consideration. If this is not done, the model's coefficients could misrepresent the true risk for future policies, leading to suboptimal pricing or risk assessment.

Reviewing the time consistency of parameters involves analyzing how coefficients evolve over different periods. This can help identify general instability or emerging trends that may otherwise go unnoticed. Some techniques to do this include:

- For each feature individually create a new GLM where that feature is interacted with time. Then either inspect the fitted coefficients or (having used LASSO regularization) check if the interaction effect is selected by the LASSO. The latter approach can easily

be automated.

- Segmented modeling. In our context, by "segmented modeling" we mean fitting the model to one year of data at the time and comparing the coefficients as time moves forward.
- Rolling window analysis. This is similar to segmented modeling except that the periods overlap. For example, with five years of data we fit a model to years 1–3 and then 2–4, etc. Finally we compare the resulting coefficients.
- Regular recalibration. This means refitting the existing model on updated data.

These (and other) methods can help us take a proactive approach to monitoring and to adjust for time inconsistencies. This, in turn, helps ensure that the models remain robust and relevant, even as underlying risk factors evolve.

10.4.2. The meaning of data can change over time

When relying on third-party models, such as obtaining a roof score[85] via API from a vendor, it is important to recognize the potential risks associated with changes to those models. If the vendor updates their model, the new scores might represent something different than historical records, which could lead to inconsistencies in your analysis. While adding a control variable for the model version might help if the vendor notifies you of changes, this approach may be insufficient on its own. Resolving these discrepancies often requires a case-by-case approach, depending on the nature of the changes and the data involved.

A relevant example in the US is the use of credit-based scores, where an actuary might be working with different programs that use varying credit models. Combining these scores or simply applying control variables may not adequately address the differences. Properly accounting for these variations often requires deep domain knowledge or additional outside analysis to ensure the integrity and consistency of the data. Without appropriate adjustments, reliance on third-party models can introduce significant risks, potentially compromising the accuracy of predictive models.

10.4.3. Business mix can change over time

Generally, GLMs are robust to mix of business changes. By this we mean that well-fitted GLMs (with appropriate feature engineering and interaction effects) can produce reliable results even when there are shifts in the distribution of policyholders,[86] such as writing relatively more young drivers in recent years compared to earlier periods. However, this

[85] For household insurance, external vendors exist that provide scores of roof material, shape, and condition that can help predict frequency or severity of future claims related to roof damage or failure. These scores are typically based on analysis of geospatial imagery.

[86] Above, we referred to model drift. The shift referred to here is known as data drift.

robustness has its limits, particularly when the changes are substantial and involve factors that were previously excluded from the model.

For instance, if underwriting rules had previously prevented the inclusion of coastal risks, but now these risks are being written, the model may struggle to accurately predict outcomes. (This is a specific example of the danger of extrapolation beyond the range of the training data.) Even with geographic variables and territorial ratemaking in place, the GLM might not fully account for the differences between coastal and non-coastal risks, leading to potential inaccuracies. The model may be attempting to fit fundamentally different risks within the same framework, which can undermine its predictive power. In such cases, additional modeling adjustments or more granular risk segmentation may be necessary to maintain accuracy.

10.4.4. When cross-validation will fail

The point above, that business mix can change over time, is very pertinent to our use of cross-validation. We have discussed that cross-validation is an estimate of generalization performance—how well our model will do on policies that will be written in the future and which it has not seen during training. Cross-validation performance is calculated by taking an average over policies in the cross-validation folds. Within those policies there may be small segments which perform really poorly but where there are not currently sufficient policies for this to be noticeable in overall performance (or to calculate coefficients accurately). If we incorrectly price those policies and attract a large number of them (due to adverse selection), our business mix will in the future be skewed towards the poorer performing segment, and actual performance will be far worse than cross-validation performance.[87] A careful check of proposed price changes by segment can help limit this risk. If possible, price comparisons by segment with chief competitors should be carried out. An ongoing and careful monitoring of business mix post rate change can be useful in allowing the business to react quickly to any issues that arise.

10.5. Regulation

In predictive modeling, it is important to recognize that the meaning of predictors can vary significantly depending on the context, particularly across different states or jurisdictions. For instance, the criteria for a safe driver discount might be determined by the underwriter in one state, while in another state, it could be mandated by the regulator to be offered to anyone who completes a specific course. This variation means that simply using the state as a control variable may not adequately capture the true differences in what "safe driver" represents from one state to another. To address this, actuaries may

[87] This issue is not unique to cross-validation—it applies more generally. Most standard modelings technique are far from impervious to the baleful confluence of thin data and consumer response.

need to employ interaction terms or residual models to better account for these material differences and improve the accuracy of their models. Another example—an insurer may have been collecting "number of stories" for a homeowners line of business. However, those data may have been populated in many different ways over the years. First, it was agent input, then a vendor provided it based on public record, now a new vendor is providing it based on satellite imagery.

Depending on the line of business and jurisdiction, regulation can also impact factor limitations. The main rating variable where the regulations limit or heavily regulate use is credit scores. In home and auto insurance, this is highly predictive but controversial in its use. Some states have outright disallowed its use in rating, while others restrict it to underwriting only (i.e., writing company placement). There are the "disallowed" rating variable (race, religion, education, income, etc.) which, despite being predictive in many cases/lines, are not allowed. These are the main restrictions; so while they may be predictive in a model, use in practice for rating, underwriting, marketing, etc., are challenged for insurance companies. We discussed in Chapter 8 the options for dealing with disallowed data types during the modeling process.

10.6. Outliers and Model Stability

Outliers in predictors, rather than in the observed target variable, can significantly impact methods such as the GLM, particularly in insurance datasets. For example, insuring a Ferrari—an extreme outlier in terms of vehicle value and performance characteristics—can skew the distribution of predictors such as vehicle cost and engine size. These outliers can exert an undue influence on the model, potentially distorting the relationships between predictors and the target variable, leading to biased or unstable coefficients. Additionally, they can inflate the variance of the model's predictions, reduce overall model accuracy, and complicate the interpretation of the results. To mitigate these effects, it is crucial to identify and appropriately handle outliers. The simplest approach is to omit such records, but alternatives include techniques such as robust regression, transformation of variables, or applying capping and flooring strategies to limit the influence of extreme predictor values.

10.7. Data Quality

Data are seldom complete or correct.

Before carrying out an analysis, we need to understand the deficits in our data. Ideally, we will ensure that our data is of a reasonable quality, and it remains so over time. This is hard. Scrubbing (making the data usable for modeling purposes) includes simple things like making sure there are no negative ages; that there are no impossibly large values of variables; that levels of categorical factors are consistent over time and that their mean-

ing is consistent over time; that where data is missing, we understand why, and so on. There may be other departments who also have examined the quality of this data in the past or who do so on an ongoing basis for various operational processes. We should take advantage of what they have found. There could be operational issues causing errors in our data. As important as remediating the data is, so too are tests for confirming that the data has been remediated and that these tests are independent of the specific remediation techniques applied.

Inevitably, we spot further errors during modeling—some data deficits are most (or only) visible during the model training or evaluation process. Ideally, we would have the time to return to data preparation and remediation during the modeling steps which follow it and to redo our models. We can help ourselves by ensuring that our modeling process is well documented and replicable, so that if rework is required, it can be done quickly.

Even after our best efforts, the statement—data is seldom complete or correct—is usually true. We are then faced with the challenge of what changes can be made to the data in order to fix deficits preventing correct analysis. This is a complete topic in its own right. Here we briefly mention some possibilities:

- Observations with nonsensical values for an important feature (e.g., age) can be deleted. If the incorrect value is for a feature which we know is not important, the value can be set to missing, and then it can be dealt with together with other missing values.
- Missing values[88] for categorical variables can be set to a missing category (e.g., "XXX").
- Missing values for numeric values can be imputed, either replacing missing values with the mean (mean imputation) or using more sophisticated methods.

Besides the above points, there are broader issues. The expression "garbage in, garbage out" typically highlights the importance of accurate and high-quality data in producing reliable model output. However, this concept also applies to the composition of the data. For instance, if a book of business has been largely adversely selected against, the data may no longer be capable of generating meaningful parameters to effectively segment risk. Adverse selection can lead to a downward spiral, where the quality of the data deteriorates to the point that even sophisticated models cannot produce reliable predictions. In such cases, "garbage in, garbage out" is more than just a cautionary phrase—it's a terminal diagnosis that requires more than basic modeling adjustments to resolve. Addressing this issue often demands a fundamental reevaluation of the data collection process, under-

[88] Dealing with missing values deserves a monograph in its own right. We note here that it is important to understand if the data is missing at random or not. If there is a pattern, i.e., a certain segment of data is always missing one or more fields—the impact at both model training stage and at prediction stage should be carefully considered.

writing practices, and even the business strategy to break the cycle of adverse selection.

Many actuaries may consider combining seemingly similar datasets, such as closed programs with open ones, or merging data following a merger or acquisition with a similar business. The aim is to increase volume, and since the data might not be homogeneous, the practitioner will add a control variable which indicates which dataset the data on each line comes from and includes that variable in the GLM. However, differences in the underlying data can lead to significant issues. Adding data from disparate sources can introduce noise and obscure meaningful patterns. Therefore, great care should be taken to thoroughly evaluate whether combining datasets is appropriate, ensuring that the integrity and predictive power of the model are maintained.

10.8. Conclusion

One can say that there are three parts to creating a rating plan:

- Preparing the data.
- Carrying out the analysis.
- Making sure that the rating plan is fit to go to market.

Data preparation is hard and time-consuming. There are many aspects to it: collection of data from business systems, preparation and collection of data assets, data storage, ensuring data quality, checking for data drift, understanding each feature, and so on. An ongoing and significant effort is needed in this area (even if it is not always appreciated).

Making sure that the rating plan is fit to go to market is also a time-consuming and vital task. In many lines of business, one small misstep in a rating plan will be taken advantage of by the segment of the market which is advantaged by the mistake, and the resulting losses may well outweigh any performance advantages gained through months of rating plan analysis. Careful comparisons between rating series and also with other rates in the market are needed. In specialty lines of business, detailed discussions with underwriters are vital.

In this monograph, we have mainly focused on the easiest and quickest[89] aspect of this process. Nonetheless, optimizing performance through our analysis is clearly a big part of what gives each business the opportunity to thrive in its marketplace. We hope that this monograph goes some way towards helping the practitioner achieve that aim.

[89] ...and possibly the most fun for actuaries!

Bibliography

Bertrand, Frederic, and Nicolas Meyer. 2018. "plsRglm" [R package], https://github.com/fbertran/plsRglm/.

Boor, Joseph. 2022. "Rebalancing the Off-Balance Factor with the Complement of Credibility." *Variance* 15 (1). https://variancejournal.org/article/29981.

Breiman, Leo, Jerome Friedman, Charles J. Stone, and R. A. Olshen. 1984. *Classification and Regression Trees.* Chapman & Hall/CRC.

Cameron, A. Colin, and Pravin K. Trivedi. 2013. *Regression Analysis of Count Data.* Cambridge University Press.

Cybenko, George. 1989. "Approximation by Superpositions of a Sigmoidal Function." *Mathematics of Control, Signals and Systems* 2 (4): 303–314. https://doi.org/10.1007/BF02551274.

Efron, Bradley, and Carl Morris. 1977. "Stein's Paradox in Statistics." *Scientific American* 236 (5): 119–127. https://efron.ckirby.su.domains/other/Article1977.pdf.

Friedman, Jerome, Trevor Hastie, Rob Tibshirani, Balasubramanian Narasimhan, Kenneth Tay, Noah Simon, Junyang Qian, and James Yang. 2021. "Package 'glmnet'." *CRAN R Repository* 595. https://cloud.r-project.org/web/packages/glmnet/index.html.

Friedman, Jerome H. 1991. "Multivariate Adaptive Regression Splines." *Annals of Statistics* 19 (1): 1–67. https://doi.org/10.1214/aos/1176347963.

Friedman, Jerome H. 2001. "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics* 29 (5): 1189–1232. https://doi.org/10.1214/aos/1013203451.

Friedman, Jerome H., Trevor Hastie, and Rob Tibshirani. 2010. "Regularization Paths for Generalized Linear Models Via Coordinate Descent." *Journal of Statistical Software* 33 (1): 1. https://doi.org/10.18637/jss.v033.i01.

Fujita, Suguru, Toyoto Tanaka, Kenji Kondo, and Hirokazu Iwasawa. 2020. "AGLM: A Hybrid Modeling Method of GLM and Data Science Techniques." In *Actuarial Colloquium Paris 2020.* https://www.institutdesactuaires.com/global/gene/link.php?doc_id=16273&fg=1.

GLUM Contributors. n.d. *Tikhonov Regularization Tutorial: Seattle-Tacoma Housing Data.* Accessed April 30, 2025. https://glum.readthedocs.io/en/latest/tutorials/regularization_housing_data/regularization_housing.html.

Goldburd, Mark, Anand Khare, Dan Tevet, and Dmitriy Guller. 2025. *Generalized Linear Models for Insurance Rating.* 2025 revision. Casualty Actuarial Society. https://www.casact.org/monograph/cas-monograph-no-5.

Hastie, Trevor, Balasubramanian Narasimhan, and Rob Tibshirani. 2025. "The Relaxed Lasso." Glmnet 4.1-9 vignette, https://glmnet.stanford.edu/articles/relax.html.

Holmes, Thomas, and Mattia Casotto. 2025. *Penalized Regression and Lasso Credibility.* 2025 revision. Casualty Actuarial Society. https://www.casact.org/publications-research/publications/flagship-publications/cas-monographs/monograph-no-13.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2021. *An Introduction to Statistical Learning: with Applications in R.* 2nd edition. Springer New York. https://doi.org/10.1007/978-1-0716-1418-1.

Kondo, Kenji. 2021. "aglm: Accurate Generalized Linear Model." R package version 0.4.0, https://CRAN.R-project.org/package=aglm.

Kuhn, Max. 2008. "Building Predictive Models in R Using the caret Package." *Journal of Statistical Software* 28 (5): 1–26. https://doi.org/10.18637/jss.v028.i05.

Kuhn, Max, and Kjell Johnson. 2019. *Feature Engineering and Selection: A Practical Approach for Predictive Models.* Chapman & Hall/CRC.

Kursa, Miron B., and Witold R. Rudnicki. 2010. "Feature Selection with the Boruta Package." *Journal of Statistical Software* 36:1–13. https://doi.org/10.18637/jss.v036.i11.

Landry, Mark. 2023. *Machine Learning with R and H2O.* H2O.ai. https://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/RBooklet.pdf.

Lindholm, Mathias, Ronald Richman, Andreas Tsanakas, and Mario V. Wüthrich. 2022. "Discrimination-Free Insurance Pricing." *ASTIN Bulletin* 52 (1): 55–89. https://doi.org/10.1017/asb.2021.23.

Lovelace, Robin, Malcolm Morgan, Layik Hama, and Mark Padgham. 2019. "stats19: A Package for Working with Open Road Crash Data." *The Journal of Open Source Software* 4 (33): 1181. https://doi.org/10.21105/joss.01181.

Mahler, Howard C. 1986. "An Actuarial Note on Credibility Parameters." *Proceedings of the Casualty Actuarial Society* 73:1–26. https://www.casact.org/sites/default/files/database/proceed_proceed86_86001.pdf.

Maitra, Saikat, and Jun Yan. 2008. "Principle Component Analysis and Partial Least Squares: Two Dimension Reduction Techniques for Regression." *CAS Discussion Paper Program* 2008:79–90. https://www.casact.org/sites/default/files/database/dpp_dpp08_08dpp76.pdf.

Meinshausen, Nicolai. 2007. "Relaxed Lasso." *Computational Statistics & Data Analysis* 52 (1): 374–393.

Micci-Barreca, Daniele. 2001. "A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems." *ACM SIGKDD Explorations Newsletter* 3 (1): 27–32. https://doi.org/10.1145/507533.507538.

Molnar, Christoph. 2020. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.* Christoph Molnar. https://christophm.github.io/interpretable-ml-book.

Pope, Devin G., and Justin R. Sydnor. 2011. "Implementing Anti-Discrimination Policies in Statistical Profiling Models." *American Economic Journal: Economic Policy* 3 (3): 206–231. http://www.jstor.org/stable/41238108.

Shi, Sheng G. 2010. "Direct Analysis of Pre-Adjusted Loss Cost, Frequency or Severity in Tweedie Models." *CAS E-Forum* Winter 2010:1–13. https://www.casact.org/sites/default/files/database/forum_10wforum_shi.pdf.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection Via the Lasso." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58 (1): 267–288.

Tibshirani, Robert, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. 2005. "Sparsity and Smoothness Via the Fused Lasso." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67 (1): 91–108.

Tobler, Waldo R. 1970. "A Computer Movie Simulating Urban Growth in the Detroit Region." *Economic Geography* 46 (sup1): 234–240. https://doi.org/10.2307/143141.

West, Brady T., Kathleen B. Welch, and Andrzej T. Galecki. 2022. *Linear Mixed Models: A Practical Guide Using Statistical Software.* Chapman & Hall/CRC.

Wood, Simon N. 2017. *Generalized Additive Models: An Introduction with R.* Chapman & Hall/CRC. https://doi.org/10.1201/9781315370279.

Yan, Jun, James Guszcza, Matthew Flynn, and Cheng-Sheng Peter Wu. 2009. "Applications of the Offset in Property-Casualty Predictive Modeling." *CAS E-Forum* Winter 2009:366–385. https://www.casact.org/sites/default/files/database/forum_09wforum_yan_et_al.pdf.

Zou, Hui. 2006. "The Adaptive Lasso and Its Oracle Properties." *Journal of the American Statistical Association* 101 (476): 1418–1429. https://doi.org/10.1198/016214506000000735.

Zou, Hui, and Trevor Hastie. 2005. "Regularization and Variable Selection Via the Elastic Net." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67 (2): 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x.

ABOUT THE SERIES:

CAS monographs are authoritative, peer-reviewed, in-depth works focusing on important topics within property and casualty actuarial practice. For more information on the CAS Monograph Series, visit the CAS website at www.casact.org.

Expertise. Insight.
Solutions.

www.casact.org