A Practical Introduction to Machine Learning Concepts for Actuaries

Alan Chalk, FIA, MSc, and Conan McMurtrie MSc

Abstract

Motivation. Supervised Learning - building predictive models based on past examples - is an important part of Machine Learning and contains a vast and ever increasing array of techniques that can be used by Actuaries alongside more traditional methods. Underlying many Supervised Learning techniques are a small number of important concepts which are also relevant to many areas of actuarial practice. In this paper we use the task of predicting aviation incident cause codes to motivate and practically demonstrate these concepts. These concepts will enable Actuaries to structure analysis pipelines to include both traditional and modern Machine Learning techniques, to correctly compare performance and to have increased confidence that predictive models used are optimal.

Keywords. Machine Learning; Supervised Learning; loss function; generalisation error; cross-validation; regularisation; feature engineering.

1. INTRODUCTION

This paper introduces the Machine Learning (ML) concepts used in Supervised Learning (building predictive models based on examples). There are a large variety of powerful and useful Supervised Learning techniques. There are a much smaller number of fundamental concepts which need to be understood and used to ensure that these techniques are applied correctly. In this paper we focus on the latter, discussing key ML principles that are relevant to many tasks that Actuaries may be involved with. We use a simple text-based task to illustrate the various ideas. Throughout, we introduce ML parlance and compare ML approaches to those used in traditional statistical and standard Actuarial work.

The key principles that we discuss are:

- The loss function
- Model evaluation measures

- Generalisation error and model validation
- Feature scaling
- Regularisation
- Feature engineering

We have chosen a text mining example to illustrate the ideas. As an exercise in Natural Language Processing (NLP) though, this paper has some glaring omissions, in particular that of using traditional NLP and more modern deep learning methods to understand the topics of each sentence. Instead, we use logistic regression, a technique that allows us to illustrate the basic ML concepts in practice. Logistic regression is a member of the family of generalised linear models, a set of models widely used by Actuaries. Hence the ideas we discuss here are immediately relevant to a large part of actuarial work.

In the interest of space we limit ourselves to Supervised Learning (SL). In Machine Learning parlance the collection of items for which we have labelled historical data are called "examples". The various pieces of information that we have to describe each example are called "features". Supervised Learning is that part of ML where the tasks involve finding relationships between features of examples (e.g. risk factors of customers) and something we would like to predict (e.g. claims frequency) and then forming predictions for new examples. Other areas of ML such as unsupervised learning and reinforcement learning, are beyond the scope of this paper.

The material covered in this paper can be found in many texts, for example, Hastie et al. [1] (chapters 2 and 7). Our treatment of the material focuses on its practical application to the kind of tasks carried out by Actuaries and we hope it will be a useful addition to the literature.

2. THE TASK

The task we use is that of predicting aviation accident cause codes from sentences that describe the causes. Though this is a text based task, most of the techniques are broadly applicable to any prediction task (e.g. pricing) and, as we will see, the nature of this task allows us to sense check our models and their predictions.

The illustrative task we have chosen is motivated by the following possible scenario. Claims handlers in an insurance company have historically coded every claim with a cause code. These cause codes are very useful for analysts. They can look at the trends in frequency or cost of claims split by the cause codes, and they can build separate pricing models for the different types of claim. They are useful for underwriters who can understand the sources of risk and for the managers of claims departments, who can predict demand for the different types of claims handling specialities. Now imagine that the claims handling system is changed. This could be a change in the IT system itself or a change in the staff that handle the claims. As a result of the change, claims are either not coded at all to a cause code or they are coded inaccurately.

The ongoing lack of information would seem to be a serious problem. We can overcome this problem if under both the old and new systems, claims narratives, the few sentences or paragraphs describing the claim, are typed into the system by the claims handlers. We can then create an algorithm which uses the claims narrative to work out the cause code and use it to identify incidents that may have been incorrectly classified and to estimate the cause codes for incidents which were not classified at all. We aim here to do exactly this, based on publicly available data for aviation accidents. The data we use comes from the National Transportation Safety Board (NTSB) database.

2.1 The NTSB Accidents Database

The National Transportation Safety Board (NTSB) is an independent Federal agency charged by the Congress with investigating every civil aviation accident in the USA. As part of fulfilling their remit, they make available detailed information about every incident they investigate. The full database can be found at the following link: `http://app.ntsb.gov/avdata/`. After investigating the accidents, the NTSB records their conclusions, which they call "Findings" or "Narratives". They express these in text form (typically in a few sentences) and in a coded form.

For our purposes the "Findings" in text form and in coded form are the only two fields of the NTSB database that we use.

The Findings codes are provided at various levels. At the highest level there are five codes:

- 01 - Aircraft, i.e., a problem with the aircraft.
- 02 - Personnel Issues (Human Error), i.e., some form of human error, typically of the pilot.
- 03 - Environmental Issues. These are often weather related or obstructions near the landing area.
- 04 - Organisational Issues. These relate to incidents which are deemed to have arisen due to deficiencies in the operational procedures of any organisation involved in the accident. This can include the company operating the aircraft or the Federal Aviation Authority.
- 05 - Not determined.

Within each of these codes there are various levels of sub-codes. For example, sub-codes for Personnel Issues include Experience/Knowledge and Action/Decision. However, we limit our task in this paper to predicting the high level codes. The types of models that would best deal with the hierarchical nature of these codes are beyond our scope in this paper.

3. EXPLORATORY DATA ANALYSIS

In this section we have a first look at the NTSB data, in order to understand the nature of our task. (The extent to which data exploration should be done before model building is not obvious, and is discussed in Section 3.5 below.)

3.1 Example Narratives

Our dataset is composed of 9,825 accident narratives covering accidents over the period 2008-2015.

An example of an accident narrative is:

> "A total loss of engine power due to the fatigue failure of a third stage turbine wheel blade."

It is fairly obvious reader that this sentence expresses a problem with the aircraft and should therefore be coded as code 01-Aircraft. Likewise the following accident narrative:

> "The fatigue failure of a tail rotor blade during an external load lift."

clearly refers to an issue with the aircraft.

On inspection of additional narratives and codes, however, it is clear that some text narratives are incorrectly coded. For example, in this accident narrative:

> "The flight instructor's failure to maintain directional control during the landing."

is also coded 01-Aircraft, whereas it should be coded as 02-Human Error.

We may also suspect that some of the accident narratives may be difficult for a computer to categorize correctly. For example:

> "The early rotation of the airplane to an angle at which the fuselage contacted the runway."

is obviously pilot error, but there is no mention of pilot or error in the narrative.

Some of the narratives are quite long. For example, the following narrative coded as an organisational issue:

> "The failure of company maintenance personnel to ensure that the airplane's nose baggage door latching mechanism was properly configured and maintained, resulting in an inadvertent opening of the nose baggage door in flight. Contributing to the accident were the lack of information and guidance available to the operator and pilot regarding procedures to follow should a baggage door open in flight and an inadvertent aerodynamic stall."

The last incident above is coded as 01-Aircraft, 02-Human Error, and 04-Organisational

Issues failures to do with the aircraft, the pilot and the procedures themselves. This demonstrates another challenge and that is, that our algorithms need to be able to predict multiple causes.

Overall, we can see that the problem might be a challenge for a human being, and certainly for an algorithm.

3.2 The Most Frequent Words

As part of our initial data exploration, we are interested in seeing the most frequent words used in the narratives for each of the cause codes. We would hope to find that different words are used to discuss the different causes. Indeed, this is what we did find. Looking only at cause codes 01-Aircraft, 02-Human Error and 03-Organisational Issues, Figure 1 shows (as expected) that the most frequent words are indeed different. Words like "engine", "loss" and "failure" are associated with code 01-Aircraft, words like "failure" and "pilot" are associated with code 02-Human Error and words like "collision" and "encounter" are associated with code 03-Environmental Issues.

This suggests one or two points for our model building process.

- A simple model based on the count of each word that occurs in the narrative should be able to achieve some predictive accuracy.
- Such a model might "get confused" by certain words. For example, the word "failure" occurs frequently in both code 01-Aircraft and code 02-Human Error.
- We might be able to mitigate the above by counting not just words, but pairs of words. We would then treat the phrases "pilot error" and "engine failure" differently. In the field of Natural Language Processing word pairs are known as bi-grams.

We immediately notice that even the simplest Exploratory Data Analysis (EDA) is influencing not only the kind of model we are intending to use, but also the features that we will use within that model (e.g. bi-grams). We note in Section 3.5

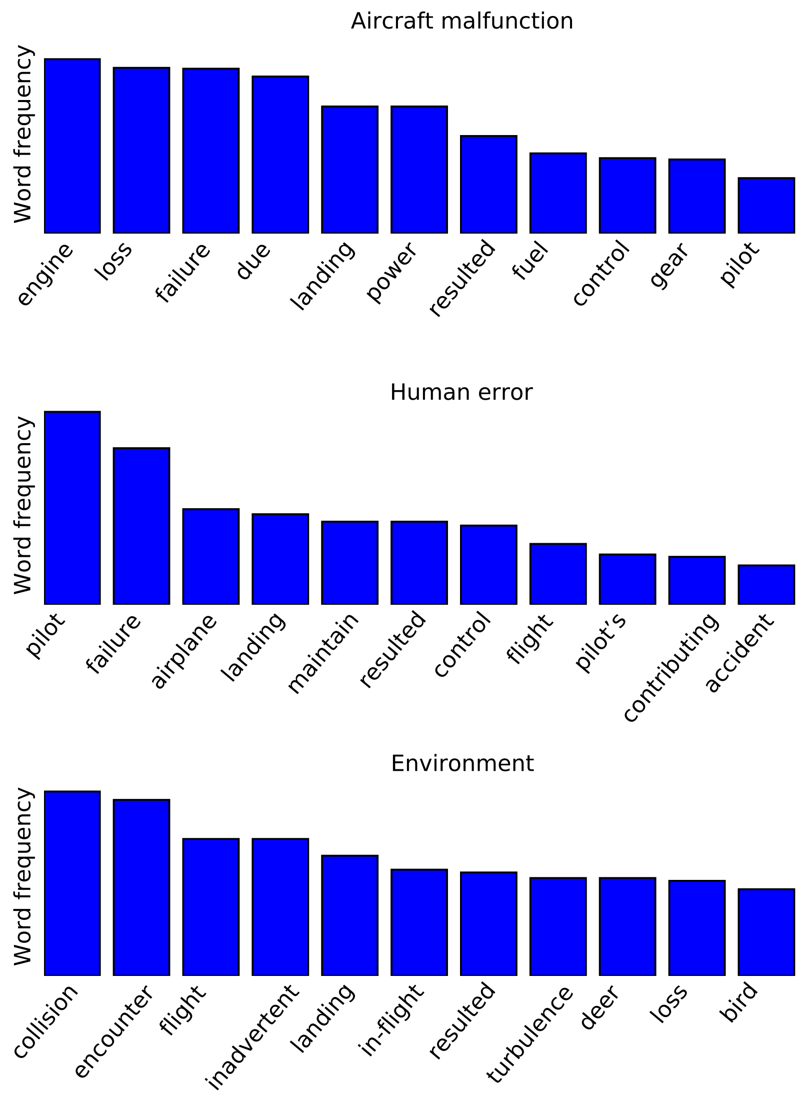

Figure 1: The words most frequently used for the different cause codes.

that overdoing the EDA is a risk. It could lead us to focus on models that suit features of the data that we happened to have noticed, and to ignore possibly much more important features that we did not notice.

3.3 Feature Engineering

When an Actuary creates a predictive model (for claims frequency, say), he will have to decide on risk factors to be included in the modelling process. These might include the age of the policyholder and the geo-location. The data which is fed into the model will typically be one record per customer per year, with each record containing all the risk factors and whether or not the customer reported any claims.

In Machine Learning parlance, each record is called an example, and the risk factors are called features. We use features to represent the example, and we seek to find a relationship (or function) between the features and whether or not the customer filed any claims. Sometimes, the features we really need are not included in the raw data. For example, in car insurance power-weight ratio may not be in the raw data. Within health insurance, height-weight ratio may not be in the raw data. If the model we are using is exceedingly flexible or powerful, it might be able to find these relationships even without us calculating them. However, for simple models such as generalised linear models, we need to calculate such features ourselves. We need to use our domain expertise and understanding of the real world to propose and calculate features. This is called feature engineering. Likewise, enriching internally gathered data with external data sources is feature engineering.

Finding and exploring new features is critical to creating good or improved models. Many hours might be spent on fine tuning a model to gain some small improvement, whereas finding a new and useful feature would provide significant improvement.

For our task in this paper, given the type of models that we fit, we need to manually find a way to represent the narratives by features so that we can fit models. The simplest way to do this is to count the occurrences of selected specific words in each narrative, and we discuss this is further detail next.

3.4 Word Counts

Possibly the simplest thing we can do is count the number of occurrences of each word in each accident narrative and then to use these counts to try to predict the cause code. To keep this manageable, we exclude very common words like "the" and "and". We then take only the most common remaining words. The ten most common words are: pilot, failure, landing, loss, control, resulted, maintain, engine, power and airplane.

As an example, consider the following narrative:

> "The pilot's spatial disorientation and loss of situational awareness. Contributing to the accident were the dark night and the task requirements of simultaneously monitoring the cockpit instruments and the other airplane."

We would represent this narrative as follows:

	pilot	failure	landing	loss	control	resulted	maintain	engine	...
count	1	0	0	1	0	0	0	0	...

Each narrative is thus represented by a set of features. The modelling problem is now similar to building a claim frequency model in a pricing exercise. For such a model we might have a few million rows of records and each row would contain certain features related to the insured, the policy and the risk and whether or not this insured had a claim. The data for a frequency model could be set out as:

	age	region	aircraft type	...	claim
policy 1	25	R1	A1	...	Yes
policy 2	30	R2	A2	...	No
⋮					

Likewise the data for our model here is laid out as:

	pilot	failure	landing	...	cause code 1	cause code 2	cause code 3
narrative 1	1	1	0	...	Yes	No	No
narrative 2	0	1	2	...	No	No	Yes
⋮							

3.5 Practical Tips

Should you explore your data before starting to build models? Actuarial work on a problem invariably includes an exploratory review of the data before starting to build models. Exploratory Data Analysis (EDA) can be split into two parts; checking that the data is correct and then a high level exploration (often visual) of relationships within the data. The latter often informs, possibly only informally or even subconsciously, some of the decisions taken during the model building process. Within the Machine Learning paradigm, it is not obvious that we should inspect the data in this way before model building. It leads to a risk that our (incorrect) thinking will "contaminate" the process of model building. After all, some ML techniques are so powerful that they can (at least theoretically) find almost any true relationship between the features and what we need to predict. The ML paradigm is to use an appropriate technique and then "let the data do the talking".

A compromise is to follow the following steps:

- Carry out a first EDA for data quality / checking purposes only.
- Fit a first set of models using automated techniques and measure the model accuracy
- Only then go back and carry out further EDA as required.

Regardless of the extent that EDA is done prior to model fitting, it is good practice that within the final model documentation, it is recorded which aspects are purely data driven and which are the result of judgement or the imposition of our prior beliefs on the model structure and parameters.

4. LOGISTIC REGRESSION

Now that we have engineered a set of features that can be used to represent each example, we can move onto our first model. We will start with a logistic regression. Logistic regression is part of the generalised linear model family and will have been used in practice by many Actuaries in building frequency models for insurance claims frequency.

As mentioned previously, one difference between our task here and other tasks, is that each narrative can have more than one label. For example, when building claims frequency models, each policy can have 0 or 1 or 2 ... claims. When described like this (rather than as a continuous frequency per unit exposure), such a problem is called "multi-class". That is to say, each policy can belong to the class of insureds that claim once, or the class that claims twice and so on. In this example each policy can belong to one and only one class. However, in our case each policy can belong to more than one class. Such a problem is called "multi-label". There are specific ways of dealing well with a multi-label problem, but because they are not broadly relevant to Actuarial work we do not use them here. Rather, we simply focus on predicting whether or not a narrative has been coded as cause code 01-Aircraft.

4.1 Results

The result of application of Logistic Regression is a scoring algorithm which can be applied to existing examples or to new examples. The score to be assigned to the claims narrative is based on the features that we fed into the Logistic Regression. In our case, these are word counts. Once a logistic regression model has been fitted, in order to decide which cause code to assign a given claims narrative, we carry out the following steps:

- Find the score for each word in the claims narrative.
- Sum all the word scores to give a score for the sentence.
- Convert the score into a "probability" that the claims narrative should be

Word	Score
failure	0.73
landing	0.10
due	0.83
gear	1.03
line	-0.74
hydraulic	3.60
extension	1.97
Sentence score	7.53

Table 1: Logistic regression scores for predicting cause code 01-Aircraft for the narrative "The failure of the hydraulic landing gear extension systems due to a ruptured line."

coded with a given cause code.

- Based on the probability, assign or do not assign the cause code to the claims narrative.

As an example, after having trained a model to predict cause code 01-Aircraft (a problem with the aircraft) based on the top 500 most frequently occurring words, consider classifying the following narrative:

> "The failure of the hydraulic landing gear extension systems due to a ruptured line."

The scores for each word and for the whole narrative are shown in Table 1.

The score for the whole narrative is seen to be 7.53. To convert this into a probability, logistic regression uses the logistic function. The logistic function of x is

$$\frac{1}{1+\exp[-x]}$$

.

The logistic function of the 7.53 score for the claims narrative is therefore

$$\frac{1}{1+\exp[-7.53]} = 0.999$$

Assuming for now that we will classify any narrative with a probability of more than 0.50 as cause code 01, we do indeed classify this narrative as a problem with the aircraft.

4.2 Interpretation

Many ML techniques result in models which are not easily interpretable. By this we mean that, if an interested party were to ask for the exact formula to use to make future predictions, it would be difficult to give the answer in a simple form. For certain applications this can matter a lot. For example, we might predict a medication to be efficacious for one patient and not for another, and, when we were challenged as to which features drive this conclusion and the magnitude of the effect of each feature, we would be unable to give a simple answer. Medical professionals might find it difficult to take decisions based on the output of such a model. In an insurance context, a model which predicts that a customer should have a price increase, but leaves an underwriter unable to understand exactly why this is the case, may be difficult for the underwriter to implement in practice . (We will actually see later that often, with some effort, even the more complex ML models can be interpreted to some extent.)

The predictions given by a logistic regression can be easily understood, and hence, we can gain insight into what words will drive the prediction of a future example. For example, looking at words with the highest absolute score gives us an idea of which words will cause a sentence to be classified one way or the other. The ten words with the largest (absolute) scores for predicting cause codes 01, 02 and 03 are shown in Table 2. We would have expected that sentences with words like "turbine", "oil" and "cylinder" would lead us to predict that a narrative is cause 01-Aircraft, and that is indeed the case.

Words having a high absolute score are not necessarily the most important words for classification of future examples. Indeed, the careful reader may have been surprised that the word "pilot" is not present in cause 02-Human Error above. The above words may have high scores, but they may not occur very frequently. The scores for the ten most frequently occurring words are shown in Table 6. We see that indeed the word "pilot" does have quite a high score in the model predicting

	01-Aircraft		02-Human Error		03-Environment	
	word	score	word	score	word	score
1	rod	10.77	spin	18.31	dark	13.36
2	lock	10.23	distraction	14.18	bird	10.56
3	oil	10.21	federal	-13.57	winds	10.31
4	trim	10.11	delay	13.55	deer	9.79
5	turbine	10.02	controller	12.57	tailwheel	-8.30
6	throttle	9.98	mechanic	11.79	testing	-7.63
7	design	9.54	distracted	11.00	cracking	-7.57
8	cylinder	9.54	see	10.99	actions	-7.44
9	gross	9.07	medical	10.72	pin	-7.28
10	door	8.76	recent	10.39	model	-5.75

Table 2: The words with highest absolute scores from the three fitted logistic regression models. The meaning of these words and why they would indicate a particular cause is mostly clear. We note, though, these words do not necessarily occur very often.

cause 02-Human Error, but not in the other two models.

	cause 01-Aircraft	cause 02-Human Error	cause 03-Environment
pilot	-0.01	2.89	-0.22
failure	0.73	0.51	0.04
landing	0.10	0.43	0.14
loss	0.27	0.34	0.28
control	0.52	0.78	-0.39
resulted	0.27	0.24	-0.05
maintain	0.54	0.72	0.19
engine	0.33	0.28	-0.58
power	-0.16	-0.58	0.41

Table 3: Scores for the 10 most frequently occurring words.

Hence, we see that although the predictions of a logistic regression are easily interpretable, we still need to take care in how we use our interpretation.

Reviewing the errors made by these models is instructive. Consider the following narrative which is coded as cause 02-Human Error (typically pilot error), but which was not predicted as such by the model:

> "The pilot's failure to maintain airspeed and aircraft control, resulting in an aerodynamic stall."

This is obviously pilot error as it contains the phrase "pilot's failure", yet the logistic regression fails to classify it correctly. The reason for this misclassification is that the word "failure" is more often associated with the failure of a part of the aircraft than it is with pilot failure. Were we to represent the document not just by word counts but by counts of phrases of two words (bi-grams) such as "pilot's failure", we should expect to see some improvement in accuracy. The process of considering which extra features to calculate and to add to the model input is called "feature engineering".

4.3 How It Works

Each time we use a model, we should try to understand how it works. This is especially important for certain Machine Learning models where it is not obvious at first sight what the model is doing. Logistic regression is not what we might call a Machine Learning model. It has a long history in traditional statistics and is part of the generalised linear model family. Nonetheless, we provide some brief comments below.

Consider a logistic regression model used to find the true probability, p, of an insured customer having an accident. In order to create a useful model, we relate p for each customer to the features of that customer. The features might be things like age, vehicle type and so on. If we are using n features in our model, then for customer j we will refer to his features as:

$$x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)}.$$

We need to convert these features into our estimate of $p^{(j)}$, the true risk for customer j. We will do this by finding a coefficient for each feature,

$$\beta_1, \beta_2, \ldots, \beta_j.$$

These coefficients are the scores we referred to in subsection 4.1 above.

Finding these coefficients is hard (see Section 5) , but once it is done, the process for finding p is exactly the same as our discussion in subsection 4.1. We first find

$$\eta^{(j)} = \beta_1 x_1^{(j)} + \beta_2 x_2^{(j)} + \cdots + \beta_j x_n^{(j)}.$$

Then we say that

$$p^{(j)} = \text{logistic}[\eta^{(j)}].$$

In our example in subsection 4.1, η worked out to 7.53 and then we had

$$\text{logistic}(7.53) = \frac{1}{1 + \exp[-7.53]} = 0.999.$$

Using the logistic function ensures that output of the model is between 0 and 1. The main work of this model is done by finding

$$\eta^{(j)} = \beta_1 x_1^{(j)} + \beta_2 x_2^{(j)} + \cdots + \beta_j x_n^{(j)}.$$

If $\eta^{(j)}$ is large, the estimated $p^{(j)}$ will be large, and, if used for classification, the example is classified as 1 or "Yes". If $\eta^{(j)}$ is low, the estimated $p^{(j)}$ will be small, and, if used for classification, the example is classified as 0 or "No". $\eta^{(j)}$ depends on the β's in a linear way. Hence, the model is just a simple linear model at heart and really only gives us access to simple ways to find relationships between features of the examples and things we wish to predict about the examples. In our task this might be not such a limitation since the value for most of the features is only 0 or 1 most of the time and, therefore, the relationship for any one feature can be expressed in a linear way. However, more generally, it should not be surprising if models that can express non-linear relationships can do bettter than "vanilla" logistic regression.

4.4 Practical Tips

- Reproducibility. Logistic regression is not an equation that can be solved with a simple formula. When using some software, a very slightly different answer may result each time the logistic regression is run, even when using the same data. It is often a good idea to specifically set the random seed used by the software to ensure reproducibility.

- Check the default settings of your software. Regularisation, an idea we will discuss in a following section, is such an obvious thing to do, that some software now do it by default. In particular sci-kit learn, the Python module used for this project will carry out some form of regularisation by default and we needed to ensure that it was "turned off", in order to produce the results in this section.

4.5 Summary and Next Steps

So far we have a seen a simple logistic regression model. There are, however, many unanswered questions:

- How good is our model?
- Even if we know our model is good for existing examples, how do we know if our model is any good for future examples?
- We used the 500 most frequently occurring words? We could equally as well only have used the top 10 occurring words or the top 2000. How do we know which is best?
- We saw that some words occur very frequently and some occur far less frequently. This means that the average word count will be much higher for some words than for others. Does this matter? Can we do anything to check?
- Can we add other features to our model which will help improve predictive power?

To answer these questions, we will need to discuss various key areas of Machine Learning practice:

- Model evaluation measures (Section 6)

- Generalisation and model validation (Section 7)
- Feature scaling (Section 8)
- Regularisation (Section 9)
- Feature engineering (Section 10)

Before discussing model evaluation, we consider another critical aspect of Machine Learning thought, the loss function.

5. LOSS FUNCTIONS

The idea of telling the machine exactly what matters and then letting the machine automatically find the best model within a class of models is fundamental to Machine Learning thought. We tell the machine "what matters" by defining a "loss function" which is high when the fitted model is "bad" in some way and "low" when the fitted model is good. We then simply have a minimisation problem: we need to minimise the loss function. (The meaning of loss function in this paper should not be confused with its use in actuarial work where it is sometimes taken to mean the distribution function for claims severity.)

There is often a duality between looking at predictive modelling as a task in maximising some kind of statistical likelihood or probability on the one hand, or treating the task as loss function minimisation on the other hand. In this section we demonstrate this duality for the logistic regression model discussed in Section 4. In other words, solving the problem of logistic regression from a maximum likelihood perspective gives exactly the same result as treating the problem as an exercise in minimising the sum of errors where the value of each error is defined by a certain loss function. Why bother with loss functions when maximum likelihood gives the same answer? It turns out that they provide flexibility in designing powerful ranges of models. This flexibility is not as easily available under the probabilistic approach.

5.1 The Maximum Likelihood Approach

Let us start with the maximum likelihood approach to logistic regression. We will show how this works from first principles. (Reading this section is not critical for the flow of the paper, but will give useful insight into the link between traditional approaches and more general ML techniques.)

We step away from our text prediction task for a moment and consider a model to predict whether or not a customer will make an insurance claim. A traditional view of logistic regression is as follows: for any customer we are looking at, let "Claim" be a random variable which takes value 1 if that customer ends up making a claim and 0 if not. Let p be the probability that the customer will make a claim, i.e., $\Pr[\text{Claim} = 1] = p$. In other words, "Claim" follows a Bernoulli distribution with probability p. If we had to guess in advance whether a customer would make a claim and we knew the value of p, we would guess "yes" if $p > 0.5$. However, we don't know p for new customers. In fact, we don't even know the value of p for existing customers, we only know whether or not they made a claim. It is possible that p, the true risk of a customer making a claim, is 0.99, yet they were fortunate and did not have an incident in the past year.

We saw in subsection 4.3 that solving a logistic regression requires us to find the best set of βs where:

$$\eta^{(j)} = \beta_1 x_1^{(j)} + \beta_2 x_2^{(j)} + \cdots + \beta_j x_n^{(j)} \tag{1}$$

and

$$p^{(j)} = \text{logistic}[\eta^{(j)}]. \tag{2}$$

The traditional statistical way is to approach this from a probabilistic perspective. We certainly know whether each existing customer has claimed or not. Let $a^{(j)} = 1$ if customer j claimed and 0 otherwise. Assume for now that we know $p^{(j)}$ for customer j. If they did indeed claim, then the likelihood of this is $p^{(j)}$. Since in this case, $a^{(j)} = 1$, we can express this as:

$$(p^{(j)})^{a^{(j)}} = (p^{(j)})^1 = p^{(j)}.$$

Likewise if they did not claim, then the likelihood of this is $1 - p^{(j)}$. Since in this case, $a^{(j)} = 0$, we can express this as:

$$(1 - p^{(j)})^{(1-a^{(j)})} = (1 - p^{(j)})^{(1-0)} = (1 - p^{(j)})^{1} = 1 - p^{(j)}.$$

So, regardless of whether the customer claimed or not, we can write this expression more generally as:

$$(p^{(j)})^{a^{(j)}} \times (1 - p^{(j)})^{(1-a^{(j)})}$$

because, if the customer did claim, then the first part of the expression works out correctly and the second part is just 1 and vice versa.

Our aim is then to find the set of $p^{(j)}$ such that the likelihood is maximised across all customers. Assuming that each customers' claims experience is independent, the likelihood for all our data is:

$$L(\beta) = \prod_{j=1}^{m} (p^{(j)})^{a^{(j)}} \times (1 - p^{(j)})^{(1-a^{(j)})}.$$

Note that we cannot manipulate the $p^{(j)}$ directly since we have decided in advance (through equations 1 and 2) that each $p^{(j)}$ is defined by the features of that customer and by the βs. Since we assume that the features are fixed, all we can do to maximise the likelihood is to find the set of βs that achieves this. That is why we write the likelihood as a function of β, i.e., we write $L(\beta)$ and not $L(p)$.

As usual in this type of approach, we take logs to simplify things. We refer to the log of the likelihood as $\log(L(\beta)) = l(\beta)$ and we have

$$\begin{aligned} l(\beta) &= \log\left(\prod_{j=1}^{m} (p^{(j)})^{a^{(j)}} \times (1 - p^{(j)})^{(1-a^{(j)})}\right) \\ &= \sum_{j=1}^{m} a^{(j)} \log p^{(j)} + (1 - a^{(j)}) \log(1 - p^{(j)}) \end{aligned}$$

Hence the problem of finding the maximum likelihood solution to logistic regression is to find the βs that maximise the above expression i.e.

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} \sum_{j=1}^{m} \left[a^{(j)} \log p^{(j)} + (1 - a^{(j)}) \log(1 - p^{(j)}) \right] \tag{3}$$

where

$$p^{(j)} = \text{logistic}\left[\sum_{i=1}^{n} \beta_i x_i^{(j)}\right]$$

5.2 The Loss Function Approach

Imagine that you have two possible solutions to a problem. How do you choose one solution over the other? You need a way to evaluate each solution. One way to evaluate solutions is to calculate the predictions and to charge some kind of loss for each incorrect prediction. We sum these charges and call the sum the "loss". We can do this for every solution which is provided to us and we then simply choose the solution with the smallest loss.

If we are predicting something which can be classified as one of a number of categories, we could simply say that the contribution to the loss is 1 if the prediction is wrong and 0 if the prediction is correct. The overall loss will then simply be the total number of errors made by the predictive algorithm. For obvious reasons this is known as the zero-one loss.

Consider the zero-one loss function further. If the result of a logistic regression for a customer is that $p = 0.75$, then since $p > 0.5$ we classify that customer as Claim = 1. If in fact that customer does not claim, then the loss is 1. The same loss would occur if $p = 0.51$ or $p = 0.99$. However, given that the customer did not claim, we might instinctively feel that a predictive algorithm which found $p = 0.51$ is better than one which finds $p = 0.99$. There is a loss function, called "cross entropy", which deals well with this issue and also has theoretically useful properties. If a is the actual class (which is either 0 or 1) and p is the result of our classifier, then cross entropy is defined as follows:

$$CE(a, p) = -a \log p - (1 - a) \log(1 - p)$$

To see how this works, consider the case when $a = 0$ (i.e., the customer did not claim). Then we have:

$$CE(a = 0, p) = -0 \log p - (1 - 0) \log(1 - p) = -\log(1 - p).$$

If $p = 0.99$ then the cross entropy is:

$$CE(a = 0, p = 0.99) = -\log(1 - 0.99) = +4.6.$$

and if $p = 0.51$ then the cross entropy is:

$$CE(a = 0, p = 0.51) = -\log(1 - 0.51) = +0.71$$

Clearly the loss gets smaller as p gets closer to zero.

If we use cross-entropy as our measure of loss, then the best βs are the ones which minimize the loss across all customers, hence we need to find:

$$\begin{aligned}\hat{\beta} &= \underset{\beta}{\operatorname{argmin}} \sum_{j=1}^{m} \left[-a^{(j)} \log p^{(j)} - (1 - a^{(j)}) \log(1 - p^{(j)})\right] \\ &= \underset{\beta}{\operatorname{argmax}} \sum_{j=1}^{m} \left[a^{(j)} \log p^{(j)} + (1 - a^{(j)}) \log(1 - p^{(j)})\right] \qquad (4)\end{aligned}$$

We can now see that the equation for finding the βs using Maximum Likelihood (equation 3) is exactly the same as that for finding the βs using the cross-entropy loss function (equation 4). Many Actuaries may be used to thinking of finding the best solution to a model as a maximum likelihood problem. Understanding that this is infact identical to minimising a loss function provides a bridge to many Machine Learning techniques.

We are now ready to move on to the questions raised at the end of Section 4, and we start with a discussion of how best to evaluate model performance.

6. MODEL EVALUATION

Consider this question:

> How good, objectively, is your model?

A comprehensive answer for a claims severity model might be:

> 90% of the time my model predicts claims severity within 50% of the actual outcome and the remaining 10% of the time it is never more than 80% out.

Practitioners involved in building insurance pricing models often "know" that the generalised linear model that they fitted is the "best" in some way because it was arrived at by a process of stepwise selection, that AIC / BIC (Akaike / Bayes Information Criterion) was minimised, or some other statistical process was followed in fitting the model. However, these statements are not really helpful. The best model from amongst many bad models is not necessarily a good model. Nor are these definitions of best particularly enlightening. Therefore, thinking through what measure should be used to evaluate model performance and to compare this measure between alternative models is a useful process. In this way we can objectively measure model performance, find the best model and know whether or not it is fit for purpose.

In practice, there are various reasons that we might decide not to use the best performing model. For example, it may take a prohibitively long time to find the correct parameters for the model. It might be that even if we know the parameters of a model, when we get a new claims narrative to classify, it takes a long time to run it through the model. There is also the issue of model transparency and interpretability (see the discussion in subsection 4.2).

Despite the above, measuring model performance plays a key part in deciding which model to use. Even if the model with the best performance is not used, it can act as a benchmark so that we know how far short the models actually implemented are of best performance.

This section motivates and describes some possible model evaluation measures for our task. We then choose one measure which we use throughout the rest of our work.

6.1 Classification or Regression

The measure we use depends first and foremost on whether the thing that we wish to predict is categorical or continuous. In Machine Learning (and statistical) parlance, models which predict a continuous variable are called regression models and models which predict categorical variables are called classification models.

Moving back to our cause code example, our task is classification. The thing that we are predicting; should a narrative be coded with a given cause code, is categorical. We will focus, therefore, on performance measures for categorical models.

6.2 Accuracy

The simplest way to measure model performance is to find the proportion of predictions that the model gets right. The accuracy of the logistic regression models fitted in Section 4.1 are shown in Table 4.

Model description	code 01	code 02	code 03
Logistic regression (500 words)	81.4%	91.2%	83.2%

Table 4: Accuracy for the logistic regression models based on the top 500 frequently occurring words.

The accuracies seem quite high - around 80% or above. However, this is misleading. Consider, for example, cause code 01. The proportion of narratives with this cause code is 76%. Therefore, a naive classifier which simply predicts that every claims narrative should be cause code 01 will achieve an accuracy of 76%. In this light, the performance of 81.4% shown in Table 4 is less impressive. Such naive classifiers can be spotted if we separately consider model performance on those examples which are due to the cause code and those which are not. The accuracy of the naive classifier is 100% on those examples due to the cause code but 0% for those which are not. Performance measures exist which do take this into account, and we look at those next.

6.3 Precision and Recall

In a sample of 1,572 narratives that were not used in fitting the logistic regression, 1,180 are coded with cause code 01 and 392 are not. Those with the cause code we call positives, and those without we call negatives.

Of the 1,180 positives, 1,051 were correctly classified as positives - we call these true positives. The other 129 were incorrectly classified as negatives. We call these false negatives. In the statistical literature, false negatives are known as Type II errors.

Of the 392 negatives, 257 were correctly classified - we call these true negatives. The other 164 were incorrectly classified as cause code 01. We call these false positives. In the statistical literature, false positives are known as Type I errors.

These numbers are not very useful in their raw form, but we can turn them into performance measures. Two key metrics are derived from these figures:

- Precision: the percentage of things we classified as positive that are true positives. In our case, this is
$$\frac{1,051}{1,051+164} = 0.865.$$
- Recall: the percentage of positives that we managed to identify. In our case, this is
$$\frac{1,051}{1,051+129} = 0.891.$$

Precision tells us how much we "care" that an example has been classified as positive. In medical tests, this is crucial. It tells us how much it matters that a patient receives a positive test result. Consider a test which shows positive 100% of the time on patients which have the disease, and shows positive only 5% of the time for patients without the disease. At face value this seems like a pretty good test. However, if out of every $1,000$ patients tested only one has the disease then in total we will have about 6 positive tests of which only 1 will be a true positive. Precision is therefore $\frac{1}{1+5} = 0.17$. A patient who gets a positive test only has a relatively small chance of having the disease (albeit far higher than the population in general).

For our naive model, Recall would be 100%, i.e. we would identify all positives, but Precision would be $\frac{1,081}{1,081+392} = 0.75$. Hence our logistic regression has a better Precision than the naive model.

6.4 The F1 Score

Recall and Precision are useful measures, but sometimes it is useful to encapsulate both of them into a simple measure. A formula often used for this purpose is:

$$\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

In our case this evaluates to:

$$\frac{2 \times 0.86 \times 0.91}{0.86 + 0.91} = 0.878.$$

This is variously known as the F-measure, F-score or F1 score.

The F1 score for the naive model is:

$$\frac{2 \times 1 \times 0.75}{1 + 0.75} = 0.858$$

which is worse than our logistic regression. Hence on the (fairly arbitrary) trade-off between Precision and Recall encapsulated in the F1 score, we would prefer the logistic regression to the naive model.

6.5 Confusion Matrix

If we label every narrative with cause code 01 as "Y" and the others as "N", we can write down the results as shown in Table 5. The numbers down the diagonal are the correctly classified cases. This is known as a confusion matrix, and it can be useful for spotting problems with models, especially when there are a large number of mutually exclusive classes.

Predicted / Actual	Y	N
Y	1,051	129
N	164	228

Table 5: Confusion matrix for the logistic regression model for cause code 01.

6.6 ROC curves and AUC

Receiver Operating Characteristic (ROC) curves and their associated Area Under the Curve (AUC) measure are an alternative model performance measure. To discuss these, we consider the case of RADAR set up to identify aircraft. Flocks of birds can be mistaken for aircraft if the RADAR are set to be too sensitive. We can reduce the problem (of having many false positives arising from flocks of birds) by reducing the sensitivity of the RADAR. However we might then not identify all aircraft (i.e. we will have many false negatives). Thus, at one extreme we can reduce sensitivity to zero, in which case there will be no false positives, but there will also be no true positives because nothing at all is picked up. At the other extreme, we can set the sensitivity so that the RADAR picks up absolutely everything. Everything which is an aircraft will be picked up, but everything which is not an aircraft will also be picked up.

The sensitivity of the RADAR therefore controls a trade off between the true positive rate (the % of all positives which are correctly classified) and the false positive rate (the % of things which are not positives but which are classified as such).

We note that:

$$\text{True Positive Rate} = \frac{\text{true positives}}{\text{all positives}} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} = \text{Recall}$$

The False Positive Rate is not directly related to Precision or Recall. Rather, it is related to yet another measure, Specificity. Specificity is the percent of actual negatives which are correctly identified:

$$\text{Specificity} = \frac{\text{true negatives}}{\text{all negatives}} = \frac{\text{true negatives}}{\text{true negatives} + \text{false positives}}$$

and

$$\text{False Positive Rate} = 1 - \text{Specificity} = \frac{\text{false positives}}{\text{all negatives}}$$

Clearly, the True Positive and False Positive Rates cannot be less than 0 or more than 1. The trade off between them traces out a curve known as the Receiver Operating Characteristic (ROC) curve. The area under the ROC curve (AUC) cannot be less than 0 nor more than 1. A perfect model will have an AUC of 1. A model which is naive and no better than random guesswork will have an AUC of about 0.50.

Within our setting, and given the output of the logistic regression, instead of classifying examples as cause code 01 when $p > 0.5$, we can classify examples as cause code 01 when $p > 0.01$ or $p > 0.99$ or indeed any threshold. As we vary the threshold we trace out the ROC curve and we can also calculate the AUC. The result is shown in Figure 2.

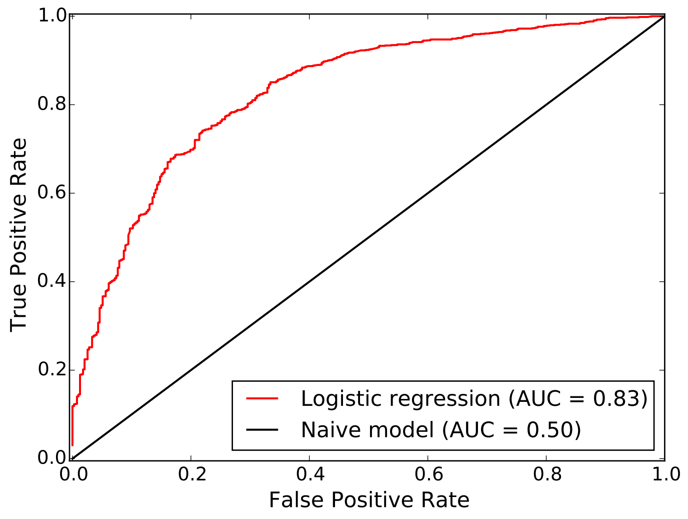

Figure 2: ROC curves for the logistic regression model fitted in Section 6 and for the naive model. We can see that the naive model is not useful for separating between examples that should be classified as cause 01 and those which should not be. On the other hand the logistic regression clearly has some power.

Whilst AUC provides a single metric which can be used to compare models, it is not at all obvious that this is a sensible measure to use in choosing models in our context. We will, after all, set the threshold at a particular value. Does it help if our model is better than other models over a range of threshold values? Although the reader may not agree with this, our point here is to illustrate that the thought process itself is important. The choice of model performance measure may affect the model that will be chosen and therefore should not be lightly undertaken (or undertaken by default).

6.7 Bespoke Measures

The most useful measure is one which is relevant to the business problem being considered. Consider insurance pricing in a market which is only marginally profitable and which is very price elastic. Any significant underpricing will bring in lots of highly unprofitable business whereas over pricing will lose top line revenue but have little impact on profit. If profit is a key performance metric, the measure used for Model Performance may need to reflect this asymmetry.

Actuarial practitioners often carry out a "decile analysis" which involves splitting the data into deciles based on the predictions output by the model ranked in increasing order. For each of these deciles, the average prediction and the average actual outcome is calculated. A plot is then made of the actual outcome (y-axis) against the predicted outcome (x-axis). If the model is "good", the resulting line will slope upwards and have a slope of roughly one. We can do this for our logistic regression and for the naive model and the results are shown in Figure 3.

This approach does demonstrate whether or not the model is achieving any predictive power. However, it can be difficult to compare between models, because it does not distill model performance to one number. Also, even if we do distil this graphic to a single number, it is not obviously related to business objectives.

As an aside, we note that actuarial practitioners often call these graphs "uplift curves". There is a form of modelling, used in marketing, personalised medicine and elsewhere, called "uplift modelling". This modelling has nothing to do with these curves, and this notation can therefore be confusing when talking to people in these fields.

6.8 Summary and Next Steps

We have looked at various model performance measures relevant to our classification task. We have noted that good practice in model building should include a

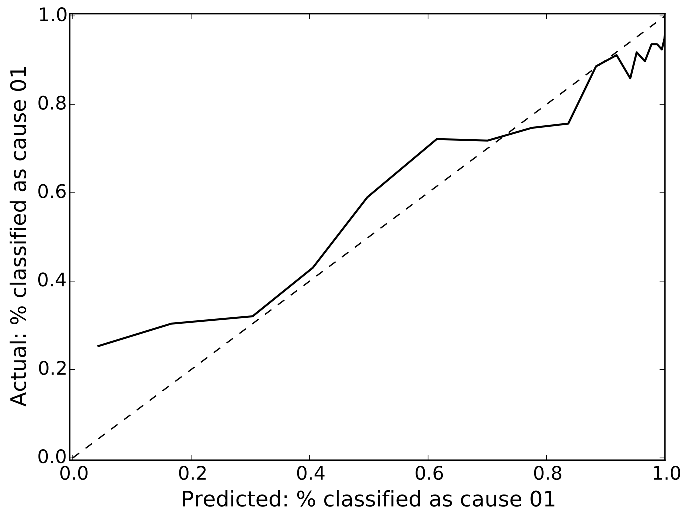

Figure 3: Decile analysis of the the logistic regression model carried out in Section 4. It can be seen that examples that the logistic regression classifies as cause 01 with a high value of p are indeed mostly cause 01. However, the set of examples classified by the logistic regression as not being cause code 01 and which have a value of p close to zero are still in fact more than 20% cause code 01.

deliberate and thoughtful choice of performance measure. For our purposes we will use Accuracy because:

- It is simple.
- It is easy to communicate; i.e., it is meaningful to say that "90% of the time, our model correctly identifies whether or not a narrative should be coded as cause 01".
- For our purposes, the cost of errors are "symmetric", i.e. it is equally bad to code a narrative as cause 01 when it is not as it is to miss coding it as cause 01 when it should be.
- There is no specific downstream process which relies on our analysis through which we could consider more bespoke model performance measures.

We have now addressed the first question raised in subsection 4.5, "How good is our model?" and we move on to the next question, "Even if we know our model is a good for existing customers, how do we know if our model is any good at all for future customers?"

7. GENERALISATION ERROR AND MODEL VALIDATION

We have fitted our model based on data on existing narratives. How do we know if our model will be any good for narratives that we have not yet seen and those that will be written in future? Given the real world context of our own task, we must immediately admit that we cannot have any guarantee at all. This is because the staff in the NTSB who write the narratives and code them could start, from tomorrow, writing the narratives in a totally different style, or coding them in a different way. They may realise that all coding to date was simply wrong and a new approach is needed. In addition, there is a more subtle risk. It could be that across some forms of sentences our model predicts very poorly, but these types of sentences are not frequently in the sample data. Something could change in the future causing these kinds of narratives to become much more common. In this case, model performance will deteriorate in practice even though NTSB staff behaviour has not changed.

Likewise, there are no guarantees in predictive modelling for pricing. Claims-making behaviour can change across the whole population or within segments. The model can be a poor predictor of frequency for a given segment, and that itself can be the cause of an increase of that type of insured so that overall, the model performs poorly and profitability reduces.

This thought process leads us towards the problems of the model refresh process and optimal timing of model refreshes. This is critical within modelling departments which may support 100's of models. We do not consider these issues in this paper inasmuch as they are not specific to Machine Learning. They exist regardless of what type of model is used.

Ignoring for now what might change in the future, how do we know if our model will perform well even if nothing changes? To be precise, we are satisfied assuming that future examples will have the same distribution of features that we currently see, and we are also satisfied assuming that the unknown relationship between features and the thing we are trying to predict remains the same. But we still wish to answer the question, how accurate can we expect our model to be for examples we have not yet seen? Generalisation and validation, two ideas which are very much part of the Machine Learning thought process, will help us answer this question, and we now discuss them in some detail.

7.1 The Train-Validation-Test Paradigm

The expected error that our model makes on future, unseen examples is called generalisation error. As we use more features and models of increasing complexity, it is possible that the relationship we find between features and what we are predicting is only due to the vagaries of the data we are looking at and will not apply to future examples.

Traditionally, Actuaries (and Statisticians) have used measures such as AIC or BIC (Akaike / Bayes Information Criterion) to control for this problem. The approach taken in Machine Learning is different and very straightforward. We divide the data that we have available to us now into two datasets. The first dataset is used for fitting the model. The second dataset is called the test set; it is put "into a vault" and not used until the end of the model building process. The first dataset is used to fit models. We can fit as many different models as we like. This may include models from different families (e.g. logistic regression, random forests, or even mixtures of the two). Finally, once we have completed our model building process, we take the test dataset "out of the vault" and calculate the errors of all our different models over the test data. We chose the model with the best generalisation error over the test data. Nowhere within our model building process (except possible in the EDA data quality checks) have we seen this data. Therefore, if our final model has a given generalisation error over the test data, we can reasonably assume that this is reflective of future model performance.

In carrying out our calculations for this paper we followed the same process. On starting our task, we put 20% of our data into the vault. We will only take it out of the vault in Section 11 in order to chose our final model and to estimate generalisation error.

We next show how the first dataset can be split into training and validation datasets in order to assist model fitting.

7.2 Model Validation, Bias and Variance

Some models are very complex and have various sets of parameters that need fine tuning. At first sight, standard logistic regression is not one of these. There is only one set of parameters - the βs, and any statistical software can easily find the best βs. We easily found the best βs when we fitted the model in Section 4. However, we had used only the top 500 most frequently occurring words and we were left with the question of how many words we should use: the top 10? the top 500? the top 5,000? If we use too few words, the model will perform poorly because it does not pick up important relationships between words and cause codes. This source of error is known as bias. If we use too many words, the fitted model will just pick up the vagaries of the sentences we are using and the fitted βs will vary greatly according to which sentences we happen to have access to. This source of error is known as variance. Hence the optimal number of words to use is a parameter which needs to be selected. Only after the parameter for the number of words has been set can a logistic regression be fit. Such parameters are known as hyper-parameters.

To help with this decision, we cannot use the test dataset. It is in the vault and can only be used to choose our final model. If we start making modelling decisions based on the test dataset, the model we fit is likely to perform better on the test dataset than on future, unknown examples. Neither can we use the dataset we are using for modelling. If we use that, we will simply find that the larger the number of words we use, the better. This is exactly the overfitting we are looking to avoid.

Instead, we split the remaining data (randomly) into a training set and a validation set. In our case, we put 80% into the training set and 20% into the validation set.

Next we fit many models on the training dataset, and we measure the accuracy on both the training dataset and the validation dataset for each model. The result is shown in Figure 4. The error on the training dataset reduces as the number of words used increases, not a surprising result. There are only 6,000 or so narratives in our training data. By the time we include 2,000 different top words in the model, we have 2,000 ways to push the fitted model towards perfectly predicting the 6,000 narratives. We can almost perfectly memorise the exact relationship seen in the sentences from the training set. However, there is no guarantee that the specific relationships we memorise will hold for unseen sentences in the validation set or for future unknown sentences. The validation error first reduces and then increases. This is also often seen. There is a trade-off between learning what is generally important and over-fitting the training data.

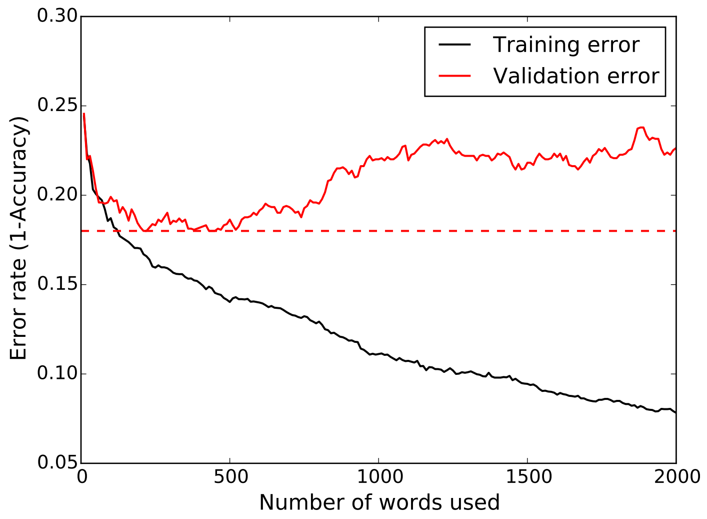

Figure 4: Training and validation curves for the number of words used. The training curve improves as model complexity increases while the validation curve follows a U shape. Too few words leads high model error due to bias, while too many words leads to high model error due to to variance.

The best validation error occurs at 200 words and then almost again at around 400-500 words. It is a little difficult to make a decision from the validation curve because it is bumpy. This is not surprising - there are only 1,600 narratives in the validation data, and therefore results are quite volatile. k-fold cross-validation is a technique to help with this volatility and we turn to it next.

7.3 Cross-validation

In the previous section we split our first dataset into training and validation datasets. This split is random and different splits would give slightly different results. This will especially be the case when the validation dataset is quite small (as for our task). Instead, we can split the first dataset into multiple (e.g. five in this discussion) parts. We take the first part and treat it as a validation set and the other four parts as our training data and we carry out the process of the previous section. We then take the second part and treat it as the validation and the other four parts as our training data and we again carry out the process of the previous section. We do this for all five parts, resulting in five validation curves. Finally, we average them to get a more stable validation curve. This is called 5-fold cross-validation. In general this technique is known as k-fold cross validation (k-fold CV). k, the number of folds, is often chosen to be five or ten. We can take this to the extreme, using the same number of folds as there are examples. Then for each fold, all the training data is used except one example. This is called Leave-Out-One Cross Validation (LOOCV).

Figure 5 compares simple cross-validation with 5-fold CV. It can be seen that the curves are similar, but the 5-fold CV curve is smoother.

It is now clear that the best validation error occurs when we use around 250 words and beyond that model performance deteriorates.

7.4 Practical Tips

Training and validation curves should look sensible. If they don't, they are probably wrong. For example, a validation curve that is below the training curve (i.e. has a lower error rate) is probably wrong.

Validation curves do not always look nicely U shaped. They can be flat, or they can reduce but not increase again. As practitioners come across such cases, they should investigate that the data, models, parameters and so on are appropriate.

A related point is whether or not we should expect the training curve to smoothly

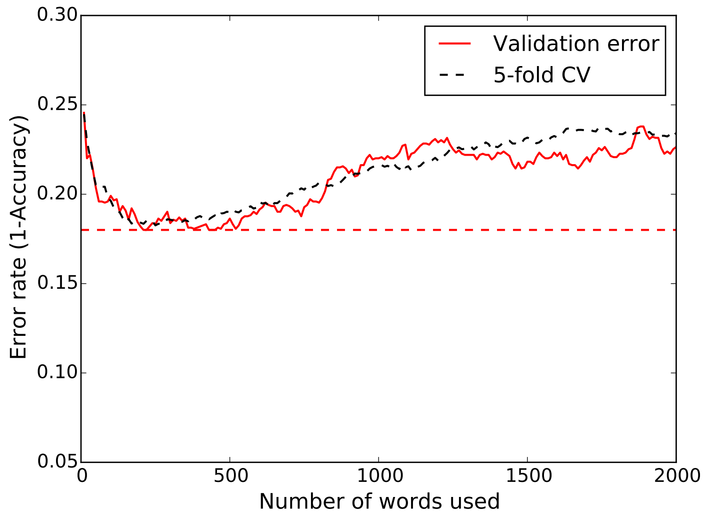

Figure 5: A comparison of cross validation and 5-fold cross validation curves. It can be seen that the they are similar, but the 5-fold CV curve is smoother.

reduce as model complexity increases. If the models are nested so that the possible solution set for more complex models includes every possible solution set of every simpler model, then we would expect the training curve to reduce smoothly.

In our case, solutions to the more complex models do indeed include all of the solutions to all of the less complex models. However the training curve in Figure 4 does not reduce perfectly smoothly. This is because we are measuring model performance based on Accuracy which is different to the measure (loss function) used for fitting (cross-entropy as discussed in Section 5). Figure 6 shows cross-entropy (scaled to be visible on the same axis as Accuracy) together with Accuracy. It can be seen that the cross-entropy does indeed reduce smoothly as model complexity increases.

Overall then, training and validation errors should “make sense”. If they don’t the reasons should be investigated.

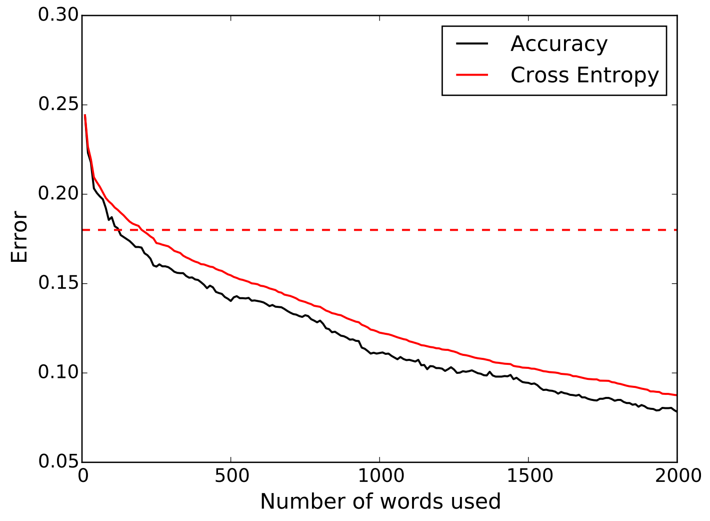

Figure 6: A comparison of cross validation curves based on Accuracy and Cross Entropy (CE). Since logistic regression minimises CE, the curve based on CE is smooth.

7.5 Summary and Next Steps

We have seen that there is no guarantee that models will perform well in the real world. This is because things may change in the real world which make our model obsolete or which expose weaknesses in our model which were not previously important or even known. We have also seen that even without such changes, there is a risk that we overfit our models to the data we have available. We discussed a solution to this which involves putting a test dataset "in the vault" and then using validation techniques on the remaining data. Using the techniques in this section have shown us that a logistic regression model based on the top 250 words is a reasonable choice for cause code 01. The 5-fold cross-validation error is 0.183 (Accuracy of 0.817). Of course we have no idea of what the Accuracy will be on the test set, but we should expect it to be slightly worse than this.

We now turn to the next question we raised in section 4.5, "We saw that some words occur very frequently and some occur far less frequently. This means that the average word count will be much higher for some words than for others. Does this matter? Can we do anything to check?"

8. FEATURE SCALING

In this section we discuss feature scaling, a data preparation method which is necessary before certain Machine Learning techniques can be used and without which they will fail to produce sensible results. At this stage we have not yet discussed such techniques. We motivate wanting to carry out this data preparation procedure from our preference to be able to compare scores.

8.1 Results

When we apply feature scaling, the classification results that we get from logistic regression will be exactly the same, but the scores for the word features will be more comparable. The results are shown in Table 6 below where the score for "pilot" is now one of the highest scores.

	without feature scaling	with feature scaling
pilot	2.89	1.44
failure	0.51	0.26
landing	0.43	0.2
loss	0.34	0.16
control	0.78	0.37
spin	17.45	1.48
distraction	13.75	1.21
federal	-15.28	-1.04
delay	17.19	1.21
controller	11.37	1.18

Table 6: Scores from the model for cause 02-Human Error, with and without feature scaling. Feature scaling leads to scores being of similar magnitude.

As mentioned above, the true motivation within Machine Learning for feature scaling is that without it, certain methods would fail. We will see more of this in Section 9.

8.2 How It Works

In subsection 4.2 we saw that scores for important words that occur infrequently are far higher than important words which occur frequently. For example, in our training data, the word "pilot" occurs 3,429 times and has a score of 2.89 when classifying cause 02-Human Error. On the other hand, the word spin occurs only 36 times and has a score of 18.31 when classifying cause 02-Human Error. Which of these two words is more important to us in classifying a general example? The score of the word "pilot" is probably high enough to tip the balance to classifying as cause 02-Human Error in the 3,429 examples where it occurs and so it is almost certainly more important. Thus, the fact that its score is so much lower could be misleading.

Is it possible to change the features in such a way that their scores more closely reflect their importance? Before adjustment, the feature for the word "pilot" is one in the 3,429 examples where it occurs and zero otherwise. On average its value is 0.545. Similarly, the average of the feature for the word "spin" is 0.006. We simply adjust each feature so that on average it has a mean of zero and standard deviation of one. We find the mean and standard deviation of each feature (across all training examples). As we have seen, the mean of the feature for the word "pilot" is 0.545. The standard deviation for the feature "pilot" is 0.499. In every example we now take the feature for the word "pilot" (which is either zero or one since there is no narrative in which the word "pilot" occurs more than once) and deduct the mean and divide by the standard deviation. If the word "pilot" was in a narrative so that the feature was one, the feature becomes:

$$\frac{1-0.545}{0.499} = 0.91.$$

Applying the same process to the feature for the word "spin" leads to the value 1 being transformed to

$$\frac{1-0.00573}{0.0755} = 13.18.$$

Applying a transformation to each of the features separately to make their magnitudes broadly similar is called feature standardisation. The particular method we used above is sometimes called Z-score normalisation. An alternative is to ensure that each feature lies between zero and one. This can be done by deducting the minimum value and dividing by the range. (Interestingly though, this would not be useful for our data.)

8.3 Practical Tips

Feature scaling is so critical to some Machine Learning procedures, it is often built in to the programs that carry out those procedures so it is not necessary to carry it out explicitly. Clearly, one needs to know which procedures do require feature scaling and whether or not it is done automatically within the code. Explicitly carrying out feature scaling and turning off any automatic scaling within the software can lead to a more transparent process.

9. REGULARISATION

Now that we have covered loss functions and feature scaling, we are able to approach a very important idea within Machine Learning, regularisation. We will motivate this topic by referring back to our discussion in Section 7 where we found that using the most frequently occurring 250 words was optimal in terms of finding a logistic regression model which did not overfit the training data and hence generalised well. Our approach was to fit models over the top 10 words, then over the top 20 words, then the top 30 and so on. After 250 words, adding extra words increased cross validation error.

This approach has an obvious weakness. Within the setting of our task, it could well be that within the top 250 words there are words which do not help the model to generalise and indeed, there could be words amongst the many less frequent words which would help generalisation. What is more problematic is that for more general tasks, there is no obvious order at all in which to add the features. What we really need is a method that is allowed to use all the features but somehow avoids becoming too complex, leading to poor generalisation. A solution to this problem, often used in Machine Learning, is called regularisation.

9.1 Results

For cause 02-Human Error, the best 5-fold cross validation error (error of 0.183, Accuracy of 0.817) was previously achieved with the top 250 most frequently occurring words. If, instead, we use the top 5,000 words, but control model complexity using a type of regularisation called L2 regularisation, the best 5-fold cross validation error is reduced to 0.173 (and Accuracy increases to 0.827). This does not guarantee that when we finally take the test dataset out of the vault, the regularised model will perform better, but an improvement of 1% for little extra modelling effort is worthwhile in our context (where the naive model achieves 75% Accuracy).

When we carry out regularisation, we only accept more complex models if they provide a reasonable improvement in model performance on the training set. The amount of performance improvement needed in order to accept a more complex model is controlled by a parameter. We will explain in further detail below that the optimal value for this parameter can be found using cross validation. Figure 7 shows the result of this search. It can be seen that as the parameter allows models of increasing complexity (increasing values on the x-axis), we first see improvement on validation error, but when the parameter allows the models to get too complex, validation error suffers.

We will also discuss below a form of regularisation called L1 regularisation. We will explain that this has the nice property that it will set many of the βs to zero. When β is zero for a feature, the word relating to that feature is not being used at all in the model. Hence, L1 regularisation selects which words are important out of the full vocabulary of words that we choose to use. When we used a vocabulary of the top 5,000 words with L1 regularisation, the number of words used in the fitted model for cause 01-Aircraft was 1,244 (though model validation error was not as good as under L2 regularisation).

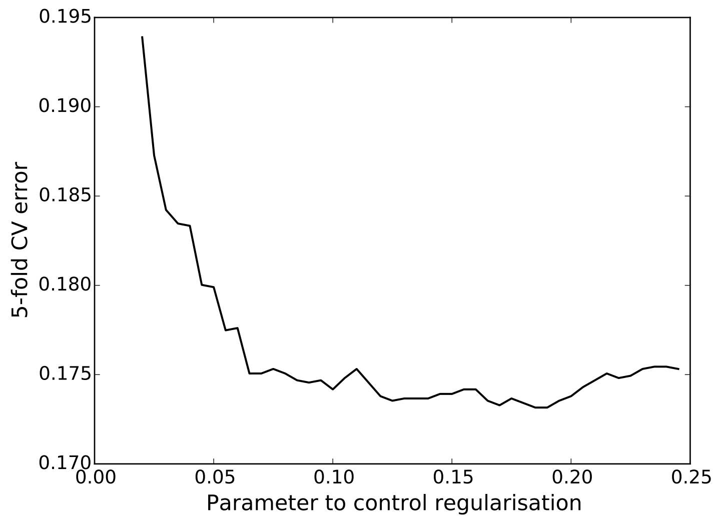

Figure 7

9.2 How It Works

As discussed previously, the power of a model to generalise (over new data) suffers when the models become overly complex and so overfits the training data. If we could tell the computer that there is a trade off between the complexity and training error, the computer would then not blindly increase model complexity just because the training error reduces. Regularisation achieves exactly this.

We have previously seen (Section 4), that the βs which are found are the result of minimising a loss function

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{j=1}^{m} \left[-a^{(j)} \log p^{(j)} - (1 - a^{(j)}) \log(1 - p^{(j)}) \right].$$

All a computer does when it solves a logistic regression, is to solve this minimisation problem. If we add to this loss function a cost for model complexity, then the solution that is found will be forced to balance training error against model complexity. This could very well provide a good solution.

How can we represent model complexity within the loss function? This is a hard question, especially since we have not defined " complex" in a way we can measure. An approach which works well in practice, follows. For logistic regression, consider the values for the βs. If there are lots of βs with non-zero values, the model is complex. We might try simply adding the values of all the βs and using this value to reflect model complexity, but then positive and negative values will cancel out. The most obvious alternatives are to add the squared values or the absolute values. If we use the sum of the squared values, the problem is now simply to find:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left[\sum_{j=1}^{m} \left[-a^{(j)} \log p^{(j)} - (1 - a^{(j)}) \log(1 - p^{(j)}) \right] + \lambda \sum_{i=1}^{n} \beta_i^2 \right].$$

where we have also introduced a new parameter λ which controls the trade-off between model complexity and training error.

Regularisation using the squares of the βs is known as L2 regularisation. Regularisation using the absolute values of the βs is called L1 regularisation and is known as the LASSO (Least Absolute Shrinkage and Selection Operator). Although we do not give the reason here, using the LASSO has the interesting outcome that many of the βs will be zero.

There is an important issue that arises when trying to implement the above. λ provides a trade-off between model complexity and training error. We decided to measure model complexity based on the size of the βs but the βs are not really unique. They depend on the scale of the features. If we measure a feature using centimetres, the β for that feature will be far smaller than if we measure that feature in kilometres. Results of regularisation, therefore, depend on the possibly arbitrary scale of the features. To deal with this, some form of feature scaling is always recommended before carrying out regularisation (unless all features are of a similar scale).

Finding solutions to regularised logistic regression is straightforward other than finding the best trade off between model complexity and training error, i.e., the best value for λ. Since λ is a hyper-parameter, as with the parameter for the number of words in Section 7, we find it using cross validation. We simply try a range of values for λ and for each we fit the regularised model on training data and find the validation error. The result of this process is shown in Figure 7 and has already been discussed above. We note, however, that the values on the x-axis are in fact $\frac{1}{\lambda}$ often referred to as C. This allows increasing values on the x-axis to represent increasing model complexity.

9.3 Practical Tips

Often, the curve produced whilst searching for the optimal model complexity will reduce quickly and then be flat over a long range. It may be that the very best cross-validation error is achieved when the model is quite complex, but a validation error almost as good is achieved earlier on using a much simpler model. Given the downsides of using more complex models, it is probably better to use the simpler model. Ad-hoc methods to chose the appropriate model complexity exist, but the main idea is to use a well reasoned approach (and not to be tempted to see performance on the test dataset).

10. FEATURE ENGINEERING

In this section we apply regularisation to help us explore further feature engineering. We create features which we think likely to improve model performance and use k-fold cross-validation, logistic regression and regularisation to decide whether or not to include the new features. We find that if too many features are added, regularisation does not find the best model. However, with some care, model performance (as measured by Accuracy using 5-fold cross validation) does improve.

10.1 n-grams

Bi-grams are phrases of two words and tri-grams are phrases of three words. Consider the narrative:

> The failure of company maintenance personnel to ensure that the airplane's nose baggage door latching mechanism was properly configured and maintained, resulting in an inadvertent opening of the nose

baggage door in flight.

In this narrative, bi-grams are, “the failure”, “failure of”, “of company”, “company maintenance”, “maintenance personnel”, “personnel to” and so on. Tri-grams are “the failure of ”, “failure of company”, “of company maintenance” and so on.

Rather than simply taking all bi-grams and tri-grams, we take only those that occur more frequently than would happen by chance given the frequency of their constituent words. For example, out of 226,340 words in the narratives, the word “go” appears 25 times and the word “around” appears 36 times. Were they to appear independently of each other, the phrase “go around” would be very unlikely to appear even once, yet it appears 22 times.

The bi-grams chosen in this manner include: go around, timely manner, aviation administration, dynamic rollover and situational awareness, which are all clearly meaningful. The tri-grams chosen in this manner include: federal aviation administration and instrument meteorological conditions and other meaningful phrases.

In the same way, quad-grams are phrases of four words and included phrases such as “loss of engine power”.

Initial results from using bi-grams and tri-grams as features were disappointing, Accuracy did not improve. Adding quad-grams to this extended feature set also showed no improvement. However when we kept the original features (based on frequently occurring words) and added quad-grams only, performance improved from an Accuracy of 82.7% to 84.1%.

10.2 Lexical Diversity

Lexical diversity measures the diversity of words used in a text. We define it here as the number of words in the text divided by the number of unique words in the text. A low value implies a large diversity of words and the minimum it can be is one. For example, in the sentence “The animal ate the animal.” there are 5 words but only 3 unique words, so the lexical diversity is $\frac{5}{3} = 1.67$.

Figure 8 shows that lexical diversity does vary between narratives according to what cause code or codes the narrative is describing. The figure shows the distribution of lexical diversity for the given categories, and it can be seen that lexical diversity is high for cause codes describing incidents due only to the environment.

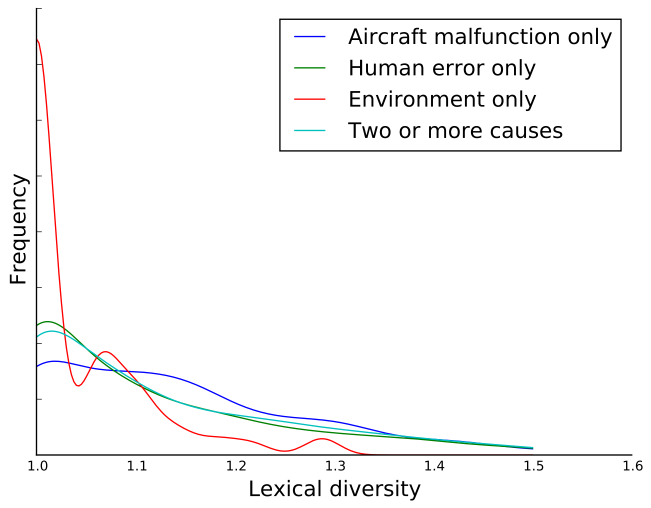

Figure 8: The distribution of lexical diversity for the given categories. It can be seen that lexical diversity is high for cause codes describing incidents due only to the environment

Adding lexical diversity to our features did not further improve accuracy, however.

10.3 Practical Tips

Regularisation searches a certain part of all possible solutions. In our work we found that mindlessly adding many features and assuming that regularisation will find the best model does not necessarily lead to a good or parsimonious model. Hence, domain expertise and some careful thought is necessary in model development.

In our task we could inspect the incorrect classifications made when using single words only and could try to understand why bi and tri-grams would not help to correct those errors. In a more general setting such as predicting claims frequency there would be no obvious way to understand why certain extra features are not helpful. Nevertheless, it is important to be honest, however much we might think that a feature should be useful. If it is not, we need to admit that there is no support for our hypothesis in the data. Incidentally, this does not mean we cannot use the feature as a rating factor, only that we should note in our model documentation that we have chosen to use something which is not supported by the data and which could, therefore, potentially harm model performance.

10.4 Summary and Next Steps

We have now covered many of the key Machine Learning ideas relevant for creating a proper structure for applying Supervised Learning techniques. Feature scaling and regularisation allow us to fit m odels w here t here m any features, whilst limiting the risk of overfitting. F eature e ngineering p rovides a w ay to improve model performance - if good features can be found. Splitting our data into training, validation and test datasets and using the test dataset only at the very end of all model fitting r educes t he r isk o f o ver-fitting an d po or model performance (generalisation error). With this structure in place, the natural next steps in our analysis are the application of techniques from traditional Natural Langauge Processing as well as more recent Machine Learning based ideas (such Gradient Boosting and Deep Neural Networks) in order to try to improve model performance. This however, is beyond the scope of this paper and instead we now conclude by taking the test data "out of the vault" and finding the model with the lowest generalisation error.

11. CONCLUSIONS

11.1 Model Comparison

Since we have been very careful not to use our test set for any part of the fitting on any of the models, we can now, finally, take our test set out of the vault and measure performance of the various models. We can be reasonably sure that, so long as recording practice at the NTSB does not change and the distribution of the types of accident do not change, the accuracy of our methods on the test set will reflect accuracy on future, as of yet unseen, examples. Table 7 shows the results.

Model description	5-fold cross validation	test
Logistic regression (top 500 words)	81.0%	81.3%
Logistic regression (top 250 words)	81.7%	81.3%
Logistic regression (5000 words and regularisation)	82.7%	83.2%
As above but 1250 words and 4-grams	84.1%	83.8%
Gradient Boosting (using trees of depth 4)	83.4%	83.5%
BoosTexter	83.8%	82.8%

Table 7: 5-fold cross-validation and test set Accuracy for the various models in this paper.

Logistic regression using the 1,250 words chosen from an earlier model together with 4-grams performs best on the test set. Although this model is more complex than some of the earlier models, the performance improvement on the test set is significant, and therefore, we chose to proceed with this model.

It is of interest to see which cases the model does not correctly classify. Over half of cases incorrectly classified would probably have been classified the same by a human being using the same narrative. For example:

> The ground crewman's failure to follow the tow bar disconnect standard operating procedures.

is classified by our predictor as not being due to cause 01-Aircraft. However, the NTSB code does flag this cause. The reason for this could be that the extended narratives, which we have not used, contain additional information. Beyond this, the models remain imperfect because they consider words such as "failure" out of context, that is, without knowing whether the narrative refers to pilot failure of a failure of some part of the aircraft. Domain specific Natural Language models or modern forms of Neural Networks would be expected to provide improved performance in this task.

11.2 Conclusions

Machine Learning techniques for Supervised Learning are very easy to apply in practice. Open-source and proprietary software make using the most modern techniques as easy as fitting a generalised linear model. However, with increased model complexity comes the risk of choosing models that will actually perform more poorly in practice.

The concepts we have discussed:

- the loss function
- choosing a model evaluation metric deliberately rather than by default
- using training, validation and test sets to correctly understand model generalisation error
- using feature engineering to enrich the data
- using feature scaling to ensure correct fitting of certain model types
- and using regularisation together with cross validation to find the best of a set of models

are key parts of the model building pipeline that is required for the fitting and choosing of appropriate models. Where not already done so, these can be easily integrated to the work flow of actuarial teams.

Acknowledgement

The authors gratefully acknowledge the the support provided by Mazars which enabled the modelling underlying this paper to be undertaken.

Biography of the Authors

Alan Chalk is a freelance Actuary with experience in predictive analytics and Data Science. He is a Fellow of the Institute of Actuaries and has Masters Degrees in Statistics and in Machine Learning. He can be contacted at: alanchalk@gmail.com.

REFERENCES

[1] Trevor J. Hastie, Robert John Tibshirani, and Jerome H. Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer series in statistics. Springer, New York, 2 edition, 2009.