

Introduction to Accurate GLM

2022 CAS RPM Virtual Seminar

Hirokazu Iwasawa and Gary Wang

March 15, 2022

Contents

Subheading here

■ Introduction	3
■ Concept	9
■ Sidebar – Flexing on the Encoding	19
■ Illustration 1 – Area	23
■ Illustration 2 – Density	32
■ Summary	39

Introduction

Reference Paper

- Suguru Fujita, Toyoto Tanaka, Kenji Kondo and Hirokazu Iwasawa
- (2020) AGLM: A Hybrid Modeling Method of GLM and Data Science Techniques
- https://www.institutdesactuaires.com/global/gene/link.php?doc_id=16273&fg=1
- Actuarial Colloquium Paris 2020

2021 Hachemeister Prize

- Commentary of the Prize Committee
 - The paper's hybrid modeling approach addresses a real need to balance the accuracy of data science techniques with the strong explanatory power of GLMs
 - The authors evaluate the advantages of the AGLM method both qualitatively and quantitatively against existing modeling methods, using an example in automobile insurance
 - The paper is accompanied by a package in the statistical software R to allow practicing actuaries to use the AGLM method easily, and
 - While the paper presents an example in pricing automobile insurance, the concepts presented may extend to other property/casualty lines or other applications.
- Overall, the committee believed this paper would be interesting to all actuaries, includes practical uses and original thought, and is well written.

History

- October 2017--- Came across the basic ideas of AGLM. A few months later, a team was set up.
- 1st half of 2019--- A satisfying beta version of a GitHub R package were accomplished.
- May 2020--- At the Online Actuarial Colloquium, a team member gave a video presentation based on a paper dated in March.
- May 2021--- It was announced that the 2021 Hachemeister Prize was awarded to the March 2020 paper.
- June 2021--- The aglm R package became available at CRAN.

In a scientific aspect

- When L1 regularization is taken, AGLM is theoretically a kind of Generalized Lasso. In this sense, AGLM can be called a “Generalized” regularized GLM approach.
- Optimization problems in such an approach have the general form:

$$\min_{\beta} -\frac{1}{n} \ell(y, X\beta) + \lambda \|h(\beta)\|$$

Here ℓ is a log-likelihood function and $\|\cdot\|$ is a norm of the Elastic Net including L1 and squared L2.

- While other approaches build specific optimization algorithms, AGLM ingeniously takes γ and X' satisfying the following identity so that optimization can be accomplished by an existing splendid regularized GLM algorithm.

$$\left(\min_{\gamma} -\frac{1}{n} \ell(y, X'\gamma) + \lambda \|\gamma\| \right) = \left(\min_{\beta} -\frac{1}{n} \ell(y, X\beta) + \lambda \|h(\beta)\| \right)$$

- But details above are never important for today's talk.
- Much more important thing is....

Vital thing

- AGLM is designed not necessarily for academic success but exactly for practitioners' availability.
- Actual practical applications will give life to AGLM.

Concepts

Main Idea

Subheading here

- The two key thrusts of Accurate GLM
 - Utilize a different encoding structure that preserves ordering information
 - Couple the encoding with variable selection through penalized regression (e.g. Lasso) to weed out insignificant breaks
- Note: This can extend to numeric variables through discretization of the variable

Set Up

Subheading here

- The GLM model

$$E[y_i] = g^{-1}(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \quad (i = 1, 2, \dots, n)$$

- Relates the numeric explanatory variables x_1 to x_p to y through the parameters β_1 to β_p
- Sets a linear relationship of the response (through the link function) to the numeric explanatory variables

Encoding Categorical Variables

Subheading here

- Classically, categorical variables are handled through one hot encoding
- Suppose we have a categorical variable x , with levels L_1 to L_k
- Define, for $x = L_i$,
 $d^U_i(x) = 1$
 $d^U_j(x) = 0$, for $j \neq i$
- We replace x with a collection of variables x_1 to x_k , with the following design matrix

x	$d^U_1(x)$	$d^U_2(x)$	$d^U_3(x)$...	$d^U_{k-1}(x)$	$d^U_k(x)$
L_1	1	0	0		0	0
L_2	0	1	0		0	0
L_3	0	0	1		0	0
...				...		
L_{k-1}	0	0	0		1	0
L_k	0	0	0		0	1

Encoding Categorical Variables

Subheading here

x	$d^U_1(x)$	$d^U_2(x)$	$d^U_3(x)$...	$d^U_{k-1}(x)$	$d^U_k(x)$
L_1	1	0	0		0	0
L_2	0	1	0		0	0
L_3	0	0	1		0	0
...				...		
L_{k-1}	0	0	0		1	0
L_k	0	0	0		0	1

- “ βx ” $\rightarrow \beta_1 d^U_1(x) + \beta_2 d^U_2(x) + \beta_3 d^U_3(x) + \dots + \beta_{k-1} d^U_{k-1}(x)$
 - As knowing $k-1$ entries of the encoding implies the k^{th} entry, we will drop the k^{th} entry in the representation as the index level
- This is the classic one hot encoding
- We will refer to $d^U_i(x)$ as the U-dummy variables (U for Usual)

Proposed Alternative for Encoding Categorical Variables

Subheading here

- We now propose the following alternative encoding
- Suppose we have a categorical variable x , with levels L_1 to L_k
- Define, for $x = L_i$,
 $d_j^o(x) = 1$, if $j > i$,
 and 0 otherwise

x	$d_1^o(x)$	$d_2^o(x)$	$d_3^o(x)$...	$d_{k-1}^o(x)$	$d_k^o(x)$
L_1	0	1	1		1	1
L_2	0	0	1		1	1
L_3	0	0	0		1	1
...				...		
L_{k-1}	0	0	0		0	1
L_k	0	0	0		0	0

Proposed Alternative for Encoding Categorical Variables

Subheading here

x	$d^o_1(x)$	$d^o_2(x)$	$d^o_3(x)$...	$d^o_{k-1}(x)$	$d^o_k(x)$
L_1	0	1	1		1	1
L_2	0	0	1		1	1
L_3	0	0	0		1	1
...				...		
L_{k-1}	0	0	0		0	1
L_k	0	0	0		0	0

- “ βx ” -> $\hat{\beta}_2 d^o_2(x) + \hat{\beta}_3 d^o_3(x) + \hat{\beta}_4 d^o_4(x) + \dots + \hat{\beta}_{k-1} d^o_{k-1}(x) + \hat{\beta}_k d^o_k(x)$
- We will refer to $d^o_i(x)$ as the O-dummy variables (O for Ordinal)
- The shifted representation is on purpose, as this sets L_k as the index level, similar as in the U-dummy variables set up

When an Encoding Level is Insignificant

If a record has level L_i , $i < k$

Under U encoding

- The model yields parameter β_i
- If $d_{i+1}^U(x)$ is deemed insignificant and dropped
- β_i is implied to be zero
- For records with level $x = L_i$
 - The model assigns 0 through the representation
 - This is the same assignment as for the base level

Under O encoding

- The model yields parameter $\hat{\beta}_{i+1} + \dots + \hat{\beta}_k$
- If $d_{i+1}^O(x)$ is deemed insignificant and dropped
- $\hat{\beta}_{i+1}$ is implied to be zero
- For records with level $x = L_i$
 - The model assigns $\hat{\beta}_{i+2} + \dots + \hat{\beta}_k$ through the representation
 - This is the same assignment as for $x = L_{i+1}$!

What the Encoding Offers, Conceptually

- Under U dummy variables, when a specific level is considered insignificant
 - Dropping the level implies grouping to base level
 - To do otherwise requires intervention
- Under O dummy variables,
 - Dropping a level implies grouping to the successive level, i.e. its subsequent neighbor in the order
- Note that, in the two encodings, each component dummy variable represents something different.
 - Make the comparison with care

Recap

- In short, the three key features of the AGLM approach for ordered and numeric variables are:
 - Discretization (for numeric variables)
 - Apply Alternative Encoding
 - Regularization
- Enough of abstract talk. Let's move on to some examples
- But first, a small sidebar



Sidebar – Flexing on the Encoding

Interpreting (and Slightly Adjusting) the Encoding

Six Level Example (which we'll use later in the illustrations)

- Consider the Ordinal Encoding

Area	Do1	Do2	Do3	Do4	Do5	Do6
A	0	1	1	1	1	1
B	0	0	1	1	1	1
C	0	0	0	1	1	1
D	0	0	0	0	1	1
E	0	0	0	0	0	1
F	0	0	0	0	0	0

Area	Do1	Do2	Do3	Do4	Do5	Do6
A	1	1	1	1	1	0
B	0	1	1	1	1	0
C	0	0	1	1	1	0
D	0	0	0	1	1	0
E	0	0	0	0	1	0
F	0	0	0	0	0	Base

- The above two mappings are equivalent mappings
 - Sets F as base
 - Beginning from E and working towards A, we cumulate Beta's to associate with the original level
 - Dropping an encoding variable, Doi, sets that level's prediction to match that of the **subsequent** level
 - Drop Do2 (in the second matrix), and B and C take on the same predictions

Transposing the Matrix

- Transposing the matrix

Area	Do1	Do2	Do3	Do4	Do5	Do6
A	0	0	0	0	0	0
B	1	0	0	0	0	0
C	1	1	0	0	0	0
D	1	1	1	0	0	0
E	1	1	1	1	0	0
F	1	1	1	1	1	0

Area	Do1	Do2	Do3	Do4	Do5	Do6
A	Base	0	0	0	0	0
B	0	1	0	0	0	0
C	0	1	1	0	0	0
D	0	1	1	1	0	0
E	0	1	1	1	1	0
F	0	1	1	1	1	1

- The similar shift gives me a more intuitive read on the intent of the matrix
 - Sets A as base
 - Beginning from B and working towards F, we cumulate Beta's to associate with the original level
 - Dropping an encoding variable, Doi, sets that level's prediction to match that of the **preceding** level
 - Drop Do2 (in the second matrix), and B and A take on the same predictions

Putting the two together

- What if we want to make a different level (other than A or F) the base? Such as level C?
 - Let's group towards C

Area	Do1	Do2	Do3	Do4	Do5	Do6
A	1	1	0	0	0	0
B	0	1	0	0	0	0
C	0	0	Base	0	0	0
D	0	0	0	1	0	0
E	0	0	0	1	1	0
F	0	0	0	1	1	1

- We can build a subsequent grouping encoding in the upper left, and a precedent grouping encoding in the lower right!
 - Dropping Do2 here, groups B into C
 - Dropping Do4 here, groups D into C
- This is the version of encoding we'll take into the exploratory illustrations



Illustration 1 – Area

Illustrative Data

- For illustrative purposes, we shall look at the dataset the paper utilized
 - freMTPL2freq data from CASdatasets package of R (Charpentier 2014)
 - French automobile insurance data
 - 678,013 records
 - 12 features
 - Consistent with the paper's treatment
 - Censored ClaimNb to 4
 - Impacts 9 records

Table 5: Features in freMTPL2freq

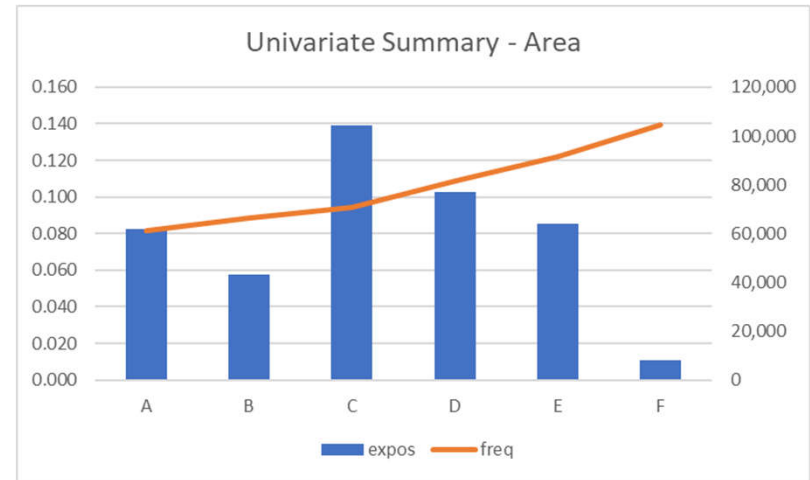
Feature	Description	Type	Ordered?
IDpol	The policy ID	Integer	Y
ClaimNb	The number of claims during the exposure period	Integer	N
Exposure	The period of exposure for a policy, in years	Real	Y
VehPower	The power of the car	Integer	Y
VehAge	The vehicle age, in years	Integer	Y
DrivAge	The driver age, in years	Integer	Y
BonusMalus	Bonus/malus, between 50 and 350: <100 means bonus, >100 means malus in France	Integer	Y
VehBrand	The car brand	Factor	N
VehGas	The car gas, Diesel or regular	Factor	N
Area	The density value of the city community where the car driver lives in: from "A" for rural area to "F" for urban center	Factor	N
Density	The density of inhabitants (number of inhabitants per square-kilometer) of the city where the car driver lives in	Real	N
Region	The policy region in France (based on the 1970-2015 classification)	Factor	N

AGLM Paper (Fujita, et. al. 2020), page 11

Explorations 1: Categorical Ordered Variable Area

- Let's start with Area as a variable
 - The information is provided as categorical levels
 - However, the information contains a sense of order
 - As we look from A to F, we are looking from rural areas towards progressively more urban areas, until F – urban centers
 - Intuitively, if level B is insignificant, we prefer that it be grouped to A or C
 - We will set C as base level, a common choice of taking the level with the most exposure as the base

Area	Rec Cnt	Expos	Clm Cnt	Freq All	Freq Train
A	103,957	61,969	5,056	0.082	0.083
B	75,459	43,012	3,800	0.088	0.088
C	191,880	104,449	9,875	0.095	0.095
D	151,596	77,120	8,390	0.109	0.109
E	137,167	63,819	7,804	0.122	0.121
F	17,954	8,129	1,131	0.139	0.137



Mapping the Area Variable

- We will set C as base level, and map through Usual encoding (Du) and Ordinal encoding (Do) as follows

```
map_Area <- function(df) {  
  df_map <- df %>%  
    mutate(Area_Du1 = case_when(Area == "A" ~ 1, TRUE ~ 0),  
           Area_Du2 = case_when(Area == "B" ~ 1, TRUE ~ 0),  
           Area_Du3 = case_when(Area == "C" ~ 1, TRUE ~ 0),  
           Area_Du4 = case_when(Area == "D" ~ 1, TRUE ~ 0),  
           Area_Du5 = case_when(Area == "E" ~ 1, TRUE ~ 0),  
           Area_Du6 = case_when(Area == "F" ~ 1, TRUE ~ 0),  
           Area_Do1 = case_when(Area %in% c("A") ~ 1, TRUE ~ 0),  
           Area_Do2 = case_when(Area %in% c("A", "B") ~ 1, TRUE ~ 0),  
           Area_Do4 = case_when(Area %in% c("D", "E", "F") ~ 1, TRUE ~ 0),  
           Area_Do5 = case_when(Area %in% c("E", "F") ~ 1, TRUE ~ 0),  
           Area_Do6 = case_when(Area %in% c("F") ~ 1, TRUE ~ 0))  
  return(df_map)  
}
```

Using Usual Encoding – GLM results

- First, let's run the GLM on the Usual encoding
- Under the U encoding

```
m_glm_u <- glm(ClaimNb ~ Area_Du1 + Area_Du2 + Area_Du4 + Area_Du5 + Area_Du6,  
              data = df_train,  
              offset = log(Exposure),  
              family = "poisson")
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	-2.35399	0.01160	-202.965	< 2e-16	***
Area_Du1	-0.13597	0.01986	-6.845	7.65e-12	***
Area_Du2	-0.07544	0.02207	-3.419	0.000628	***
Area_Du4	0.13613	0.01713	7.948	1.90e-15	***
Area_Du5	0.24049	0.01754	13.714	< 2e-16	***
Area_Du6	0.36357	0.03660	9.935	< 2e-16	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

	Area predicted
1	A 0.08291304
2	B 0.08808632
3	C 0.09498906
4	D 0.10884165
5	E 0.12081424
6	F 0.13663743

- These predictions match trained data frequencies
- If we introduce regularization and study the progression of the parameters...

Using Usual Encoding, Elastic Net (Lasso) Results

Area

- Running the same variables through Elastic Net with Alpha = 1

```
> write.csv(df_grid, "en_u_results.csv", row.names = FALSE)
> df_out <- cbind(coef(m_en_cv_u, s = 1),
+               coef(m_en_cv_u, s = .005),
+               coef(m_en_cv_u, s = .004),
+               coef(m_en_cv_u, s = .003),
+               coef(m_en_cv_u, s = .002),
+               coef(m_en_cv_u, s = .001),
+               coef(m_en_cv_u, s = 0))
> df_out
6 x 7 sparse Matrix of class "dgCMatrix"
      s1      s1      s1      s1      s1      s1      s1
(Intercept) -2.2966 -2.297646120 -2.29984587 -2.301453316 -2.313252328 -2.33331796 -2.35258308
Area_Du1     .      .      -0.01712140 -0.054943979 -0.085825890 -0.11024618 -0.13552819
Area_Du2     .      .      .      .      -0.009584672 -0.04146442 -0.07463367
Area_Du4     .      .      .      .      0.025757445 0.08120865 0.13301546
Area_Du5     .      0.005846513 0.03389413 0.075932993 0.126513030 0.18385042 0.23743718
Area_Du6     .      .      0.001719958 0.104394587 0.23959514 0.35769319
```

- Depending on the penalization, β_6 can be sufficiently repressed to create a reversal

Using Ordinal Encoding – GLM results

- Now, let's run the GLM on the Ordinal encoding
- Under the O encoding

```
m_glm_o <- glm(ClaimNb ~ Area_Do1 + Area_Do2 + Area_Do4 + Area_Do5 + Area_Do6,  
              data = df_train,  
              offset = log(Exposure),  
              family = "poisson")
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	-2.35399	0.01160	-202.965	< 2e-16	***
Area_Do1	-0.06052	0.02475	-2.446	0.014456	*
Area_Do2	-0.07544	0.02207	-3.419	0.000628	***
Area_Do4	0.13613	0.01713	7.948	1.90e-15	***
Area_Do5	0.10436	0.01822	5.728	1.01e-08	***
Area_Do6	0.12308	0.03712	3.316	0.000914	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

	Area	predicted
1	A	0.08291304
2	B	0.08808632
3	C	0.09498906
4	D	0.10884165
5	E	0.12081424
6	F	0.13663743

- Note that here, the β 's are incremental
 - For E, the sum $\beta_3 + \beta_4 = 0.240$, which compares with the β from the Usual encoding GLM model

Using Ordinal Encoding, Elastic Net (Lasso) Results

Area

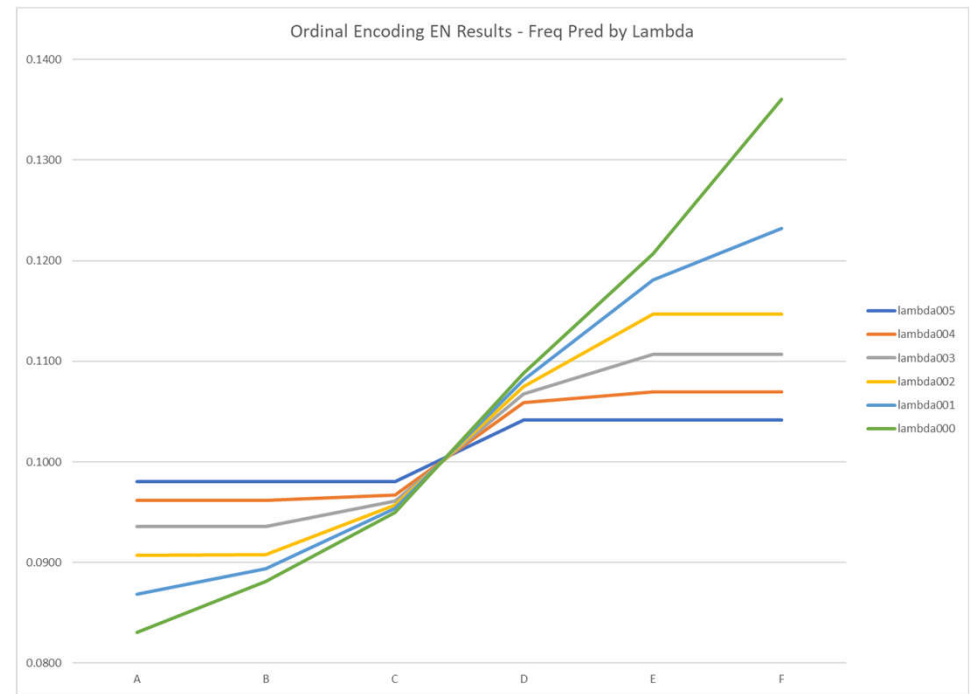
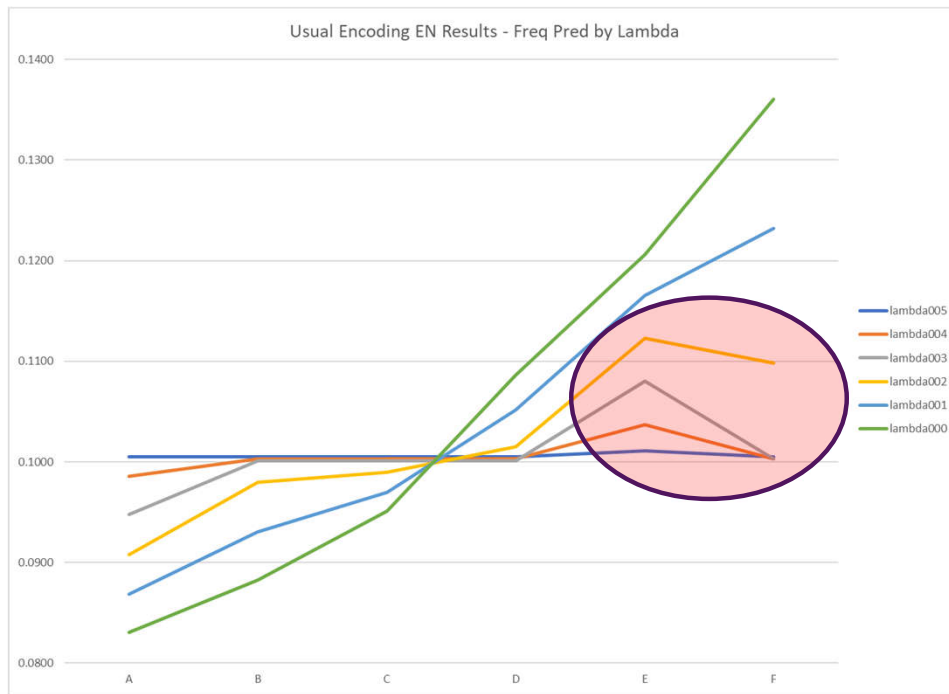
- Running the same variables through Elastic Net with Alpha = 1

```
> df_out <- cbind(coef(m_en_cv_o, s = 1),
+                 coef(m_en_cv_o, s = .005),
+                 coef(m_en_cv_o, s = .004),
+                 coef(m_en_cv_o, s = .003),
+                 coef(m_en_cv_o, s = .002),
+                 coef(m_en_cv_o, s = .001),
+                 coef(m_en_cv_o, s = 0))
> df_out
6 x 7 sparse Matrix of class "dgCMatrix"
      s1      s1      s1      s1      s1      s1      s1
(Intercept) -2.2966 -2.3221882 -2.336144001 -2.34253689 -2.3463975215 -2.35016098 -2.35384622
Area_Do1      .      .      .      .      -0.0007115925 -0.02886067 -0.05921677
Area_Do2      .      .      -0.005137390 -0.02617415 -0.0528644345 -0.06458902 -0.07492512
Area_Do4      .      0.0603331  0.091026197  0.10562796  0.1159304258  0.12601973  0.13576561
Area_Do5      .      .      0.009760542  0.03616004  0.0648469466  0.08750181  0.10362574
Area_Do6      .      .      .      .      .      0.04291759  0.11956614
```

- These parameters are incremental changes away from C
 - β_4 to β_6 are always non-negative, so we have monotonic increasing results from C to F
 - β_1 and β_2 are always non-positive, so we have monotonic decreasing results from C to F
 - Together, we have a monotonic set of predictions regardless of lambda – a good result

Progression of Elastic Net Predictions - Graphically Area

- With Usual Binning, the modeler may choose to interject and build binning of E and F
- With Ordinal Binning this is the default result, and the modeler would interject to bin away from neighbors






Illustration 2 – Density

Relation between Area and Density

Area	Density_min	Density_avg	Density_max	recnt	expos	clmct	freq
A	1	28	50	103,957	61,969	5,056	0.082
B	50	73	100	75,459	43,012	3,800	0.088
C	100	249	500	191,880	104,449	9,875	0.095
D	500	1,076	1,993	151,596	77,120	8,390	0.109
E	2,001	4,380	9,850	137,167	63,819	7,804	0.122
F	10,008	22,015	27,000	17,954	8,129	1,131	0.139

- A review of the Area and Density information confirms that Area was built up from Density data

Strategy for Developing Density Bins

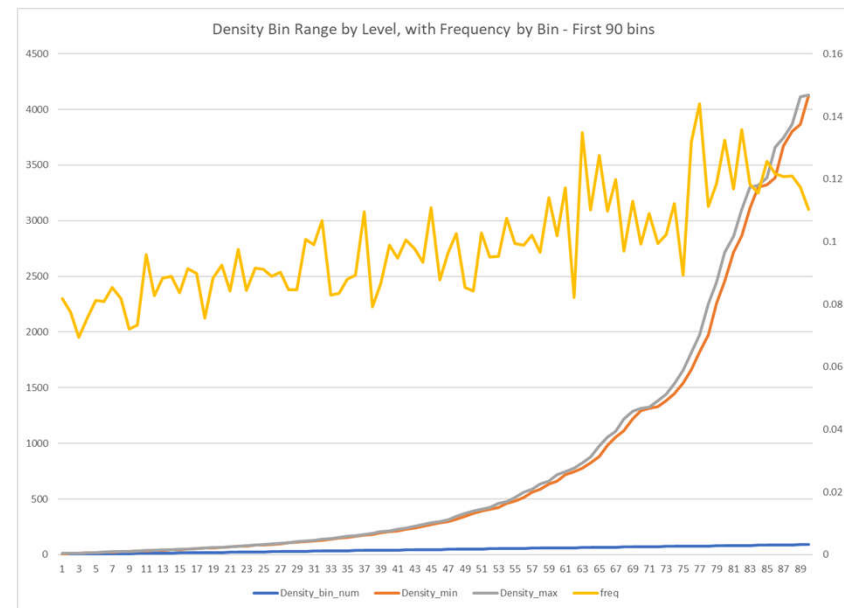
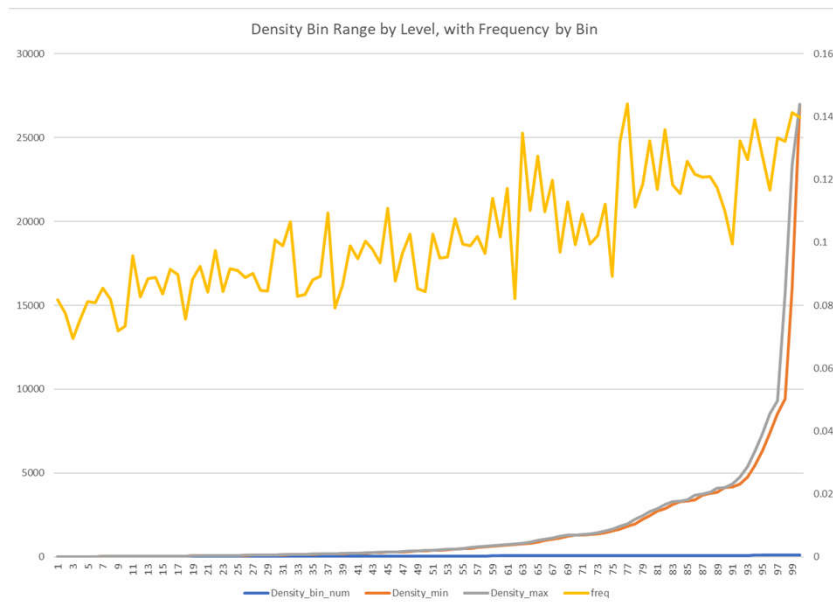
- Create Ordered dummy variables at very fine level
 - Equal sized bins (by record count or by weight)
 - Equal width bins, perhaps with tail(s)
- Feed dummy variables through regularization
- Select binning option

- We will start with 100 bins
 - By record count
 - Set first level as base
 - These are (roughly) equal sized bins, so there is less concern of picking a significantly smaller base level
 - As it turns out, the first bin is one of the largest bins available to us for use as base

Density Bin Statistics	Exposure
Minimum	1,352
Average	3,585
Maximum	5,696
Size of Bin 1: [1, 10]	5,269

Progression of Elastic Net Predictions - Graphically Area

- The density bins cover progressive wider and larger ranges of density
- The frequency appears linear in the plot, which implies it is not linear to density



Elastic Net Results – Relevant Bins

Density Bin	Bin Min Density	lambda005	lambda004	lambda003	lambda002	lambda001
Intercept		-2.3254	-2.3459	-2.3729	-2.4191	-2.4761
Do11	34	.	.	0.0003	0.0244	0.0635
Do24	84	.	.	0.0003	0.0112	0.0192
Do29	110	.	.	0.0088	0.0079	0.0063
Do30	116	.	.	0.0034	0.0070	0.0110
Do37	173	.	0.0091	0.0198	0.0252	0.0299
Do51	394	0.0136	0.0248	0.0284	0.0316	0.0354
Do54	459	0.0246	0.0271	0.0287	0.0304	0.0317
Do59	633	0.0203	0.0242	0.0272	0.0307	0.0325
Do63	778	0.0015	0.0034	0.0063	0.0086	0.0118
Do76	1,662	0.0128	0.0388	0.0651	0.0878	0.0972
Do92	4,358	.	.	.	0.0072	0.0490

- In our summary above (insignificant levels dropped from table), we have candidate binning starting with 6 levels ($\lambda = .005$)

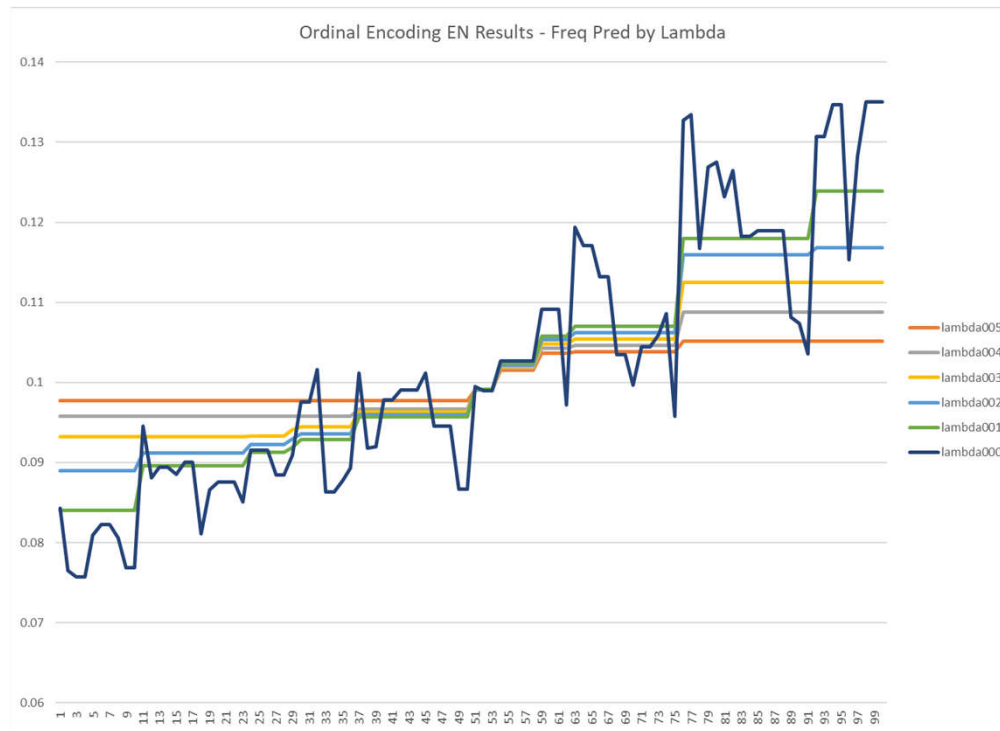
Elastic Net Results – Relativities for ($\lambda = .004$)

Density Bin	Bin Min Density	Bin Max Density	lambda005	lambda004	lambda003	lambda002	lambda001
Intercept	1	33	0.0977	0.0958	0.0932	0.0890	0.0841
Do11	34	83	0.0977	0.0958	0.0932	0.0912	0.0896
Do24	84	109	0.0977	0.0958	0.0933	0.0922	0.0913
Do29	110	115	0.0977	0.0958	0.0941	0.0930	0.0919
Do30	116	172	0.0977	0.0958	0.0944	0.0936	0.0929
Do37	173	393	0.0977	0.0966	0.0963	0.0960	0.0957
Do51	394	458	0.0991	0.0991	0.0991	0.0991	0.0992
Do54	459	632	0.1016	0.1018	0.1020	0.1021	0.1024
Do59	633	777	0.1036	0.1043	0.1048	0.1053	0.1058
Do63	778	1,661	0.1038	0.1046	0.1054	0.1062	0.1070
Do76	1,662	4,357	0.1051	0.1088	0.1125	0.1160	0.1179
Do92	4,358		0.1051	0.1088	0.1125	0.1168	0.1239

- Our method results in different binning decisions than those utilized for Area
 - We are focusing the review within this variable, without considering others
 - There may be business related reasons for the ultimate bin selections not contemplated here

Progression of Elastic Net Predictions - Graphically Density

- The Elastic Net results provide recommended binning at various lambda's, with natural groupings for unused initial bins



Summary

Summary

- The illustrated data encoding and subsequent regularization can be set up in coding environment to handle ordinal categorical variables and numeric variables in bulk
- For ordered variables (categorical, or discretized numeric)
 - Usual encoding treats insignificant levels by grouping towards base level
 - Ordinal encoding treats insignificant variables by grouping to a neighboring level
 - The ordinal treatment is a more natural default for handling ordinal variables
- For numeric variables,
 - Ordinal encoding allows for finer discretization (into more levels) initially, allowing the EN algorithm to create suitable coarser groupings
 - Usual encoding does not interact with EN algorithm in as natural a manner