Credit Scorecard Modelling

Alexander James Waudby

September 1, 2020

Abstract

Abstract Placeholder

Contents

List of Figures

Chapter 1

Credit Scoring

1.1 Introduction

Credit scoring is a method used by financial institution globally to assess whether a customer should be taken on. This can be for a variety of services such as credit cards, loans, mortgages, etc. It's development originated from the need of risk vs rewards. Lenders needed a way of determining if a potential customer would be able to pay back their credit and as such not costing the lender money by taking on creditees which end up being unable to repay the debt. A credit score is usually just a number indicating your quality as a creditee. The scale of the score can vary on which lender is providing the score but usual ones are 0-999 or 0-500 with the lower the score the less likely you would be offered the service.

Although used globally, there is no widely accepted "perfect" model or method. All companies assess their customers differently, a customer could be rejected from one lender and be accepted by another based on what they would define as an acceptable client. Even within companies the models and methods can change due to new circumstances and the changing financial climate. What previously could have been a strong predictor of a bad client could now be insignifcant. A recent example of this is the technology development of mobile phones. Previously, if a client did not have access home phone this could be an indicator of a possible bad client. Now, with the development and wide public access to mobile phones, access to a home phone has become mostly irrelevant with most of the public having no use for them anymore. Changes such as this and others require lenders to be constantly evaluating how they assess customers to prevent the rejection of good clients and the accepting of bad ones.

1.2 Home Equity Loans

The data set which will be used in the project is a set of observations of home equity loans in the US. Home equity can total to be a large portion of a US households wealth but unlike other wealth, it cannot be liquidated easily. Dispite its inability to be liquidated easily, it is a readily accepted form of collateral for credit (Canner, Durkin, and Luckett 1998) with (Weicher 1997) estimating it to account for 5-10% of the US's total mortagage origination in the 1990s. From more recent data, home equity loan origination peaked in 2005 totalling nearly $364 billion but declined during the 2008 market crash and has since, in recent years seen a steady increase (CoreLogic 2016).

Before the crisis home equity loans were popular to be piggybacked onto first mortgages with subprime rates. In recent years since the market crash companys have been more conservative when it comes to the acceptance of applicants which brings about the importance of robust credit score modelling (CoreLogic 2016).

1.3 Modelling

Credit score modelling is often discrete based, with the usual being wether a client will default or not, the most common being a logistic regression with the response variable being either a good ($y = 0$) or bad customer ($y = 1$). Predictors can be a variety of variables such as personal characteristics, age, gender or economic status e.g. car owner, home owner/rentor etc, to financial characteristics like amont of current debt and repayment statuses. One thing to note, certain personal characteristics are off limit to company due to discrimination laws. Predictors such as race, may be shown to have some use in scoring but cannot be used as the model would become discriminatory.

1.4 Outline

In this project, first I will discuss current literature on credit score modelling and other methods for data cleaning and preperation I will being using through out the project. Next I will describe the data used for the credit scoring and how I decided to prepare it, following from that I present some exploratory data analysis of each variable and my assumptions of their significance and characteritics. I will be using weight of evidence, explained in Section (2.2), to create binned variables. The next chapter will cover the results of the modelling methods applied and their comparisons and performance. Finally, I will be covering an alternative use of scorecard modelling using the example of Covid-19 and developing a model for the health risk of a patient.

Chapter 2

Literature Review

2.1 Cut-off

A scorecard in simple terms is just a method producing a score for each individual. To put the scorecard into use the difference between the scores needs to be classified this is done by a cut-off score. This score is a point on the scorecard which would seperate accepted applicants from rejected. A simple cut-off method would be to have a single score, any applicants above the score are accepted and anyone below the score is rejected. The benefit of a simple method is the ability to quickly process applicants and move desired applicants onto the next stage faster. The issue with the single cutoff comes with the applicants that are close to the cutoff, having a strict cut-off can cause a company to take on bad applicants or reject good applicants where futher investigation would prove the applicant more likely to be the opposite.

An alternative to this would be a two score cut-off. This would be done by having two scores like Rejected $< S_1 <$ Refer $< S_2 <$ Accepted. Any score above S_2 is automatically accepted and any below S_1 is rejected. Scores which the land in between and moved to a referral stage where a lender can further look into the applicants case by case to decide the outcome. This comes with added benefit of removing the issue of applicants close to the single cutoff. The idea is that with the lenders insight, more good applicants will be accepted and more bads rejected compared to the single cutoff, thus possibly reducing the bad rate of accepted applicants.

The cut-off scores can be determined by varying factors which can change depending on the companies interest. Four of these are specified by (Bailey 2004). Acceptance rate, the percentage of all applicants accepted by the cut-off. Overall bad rate, the percentage of all accepted applicants that end up being bads. Marginal bad rate, the percentage of accepted applicancts that are bad close to the cut-off score. Profitability, the possible profit from goods minus the loss from bads. Depending on the situation of the business and its goals would determine the importance of each factor with overal bad rate being the usual priority.

2.2 Weight of evidence and Information value

Weight of evidence, WOE. Is a popular method used in score card modelling, often used because the variables used in credit scoring can have a large amount of categories which cause impractiallities when converting these to dummy variables. WOE is an alternative to that, rather than creating a large amount of dummy variables, the method produces a numerical value (weight of evidence) for each category/bin which is prdouced by Equation (2.1). These values would then replace their respective categorical/bin value before using whichever modelling method used to develop the scorecard.

$$WOE = \ln \frac{f(X = x|y = 0)}{f(X = x|y = 1)} \tag{2.1}$$

where $f()$ is the distribution of category X for either goods ($y = 0$) or bads ($y = 1$).

The information value, IV. Is a measure of the weight of evidence for categories $IV \geq 0$. A value of 0 indicates the variable has no predictive power i.e. no valuable information in the variable. IV can be calculated by Equation (2.1). A guideline produced by (Bailey 2004) is below for evaluating the IV values.

$$\text{IV} = \sum (\text{\% of Bad} - \text{\% of Good}) \cdot \text{WOE} \tag{2.2}$$

IV	Recommendation
Less than 0.03	Poor Predictor
From 0.03 to less than 0.10	Weak Predictor
From 0.10 to less than 0.30	Average Predictor
From 0.30 to less than 0.50	Strong Predictor
Over 0.50	Very Strong Predictor

Table 2.1: Information Value Table
(Bailey 2004)

2.3 Performance Evaulation

ROC and AUC

ROC, Receiver Operating Characteristic. Was a method of analysis developed during World War II under "Signal Detection Theory". It was originally used for radar operators and their ability to determine if a blip on screen was an enemy or just noise, hence the name Receiver Operating Characteristics (Tape 2000). Since then, the method has been applied into a variety of fields for visuallising the accuary of classification models.

Understanding the ROC Curve is relatively simple, the plot is the false positive rate against true positive rate for different cutoff points. The true positive rate is seen as the sensitivity and the false positive being (1 - specificity) An example figure can be found below, the higher the curve, the more accurate the model can be seen as, with the neutral line going 45 degrees through the plot can be seen as the model being the same as a 50/50 guess on the outcome. In some cases these curves can overlap and cause some ambiguity on which curve is overall the best so the measure used to remove this amiguity is the AUC, Area under the curve (2.3). A higher AUC inidicates a stronger disciminatory power with 0.5 being none and 1 being a "perfect model". As such the model with a higher AUC can be considered "a better model". Generally, an $AUC > 0.8$ is considerd good.

$$A = \int_c F_1(c) {F_0}'(c) dc \tag{2.3}$$

A more common representation of the AUC is the gini coefficient (2.4). A linear transformation of the AUC to allow the measure to have a preferred 0 to 1 scale rather than 0.5 to 1.

$$gini = (2 \cdot AUC) - 1 \tag{2.4}$$

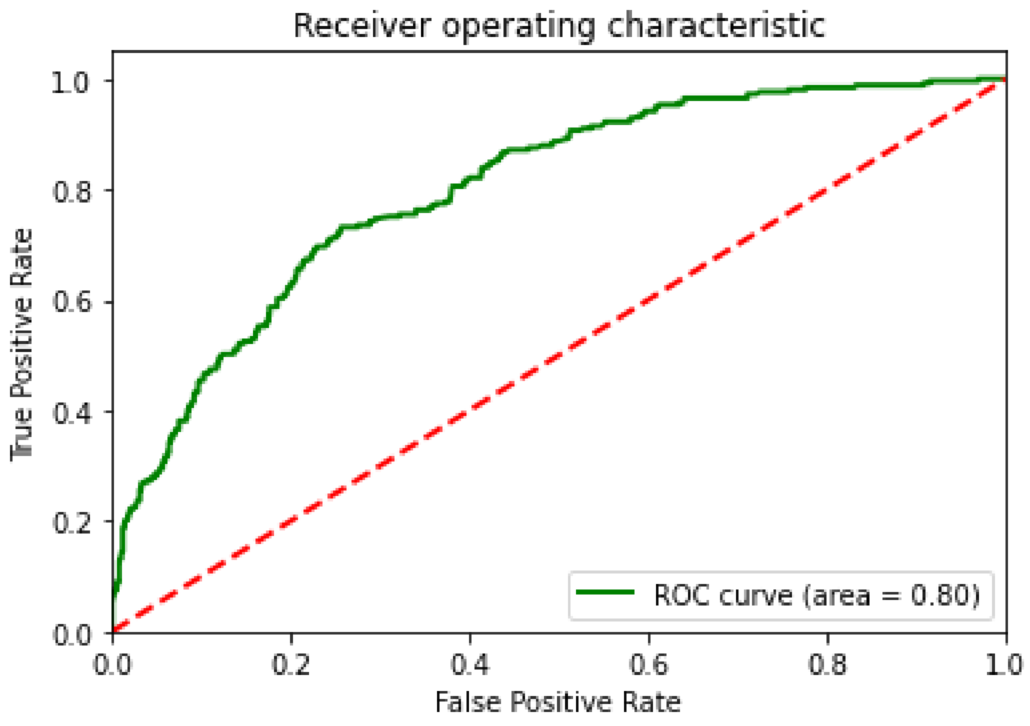

Figure 2.1: ROC Example

K-S Statistic

The K-S Statistic (Kolmogorov-Smirnov Statistic) is a measurement of the scorecards ability to seperate the goods from bads. The K-S Statistic is the maximum distance between the cumulative distributions of both the goods and bads, or alterntively, if $F_g(x)$ is the cumulative distribution of goods and bads is $F_b(x)$ where x is the score then the K-S Statistic is (2.5)

$$KS = max(F_g(x) - F_b(x)) \tag{2.5}$$

An issue of this measurement is that it only provides the score at which the scordcard seperates the goods and bads the most. The cutoff score for the card might not necessarily be this score and a higher K-S score does not imply the scorecard is a better fit.

Divergence

Divergence is a measurement of the distributions of goods and bads. The idea is that the scorecard on average will assign a lower score to bads than goods i.e. $\mu_b < \mu_g$. Divergence is a way to assess this performance. Specified by (Bailey 2004) divergence is calculated by Equation (2.6)

$$Divergence = \frac{(\mu_g - \mu_b)^2}{\frac{1}{2}(\sigma_g^2 + \sigma_b^2)} \tag{2.6}$$

Where μ is the mean and σ^2 is the variance and g and b are goods and bads respectively

Population Stability Index

The population stability index, PSI. Is a measure of the distributions of two populations to ensure similarity. Credit score models are developed using historical data and it is important to ensure that the data used to model the scorecard does not differ too much from the data the model will be used to assess. A high PSI can result in investigation as the use of the model could potentially be unsuitable and cause an increase in risk from the company (Yurdakul 2018). Although the data being used is collected in the same time period, PSI can still be used to compare train and test data sets to ensure there isn't a large population shift between the two.

Interpretation of the PSI in the industry is not set in stone. The general guideline mentioned by (Bailey 2004) is that a psi of less than 10% indicates no shift, 10 to 25% shows a slight shift and should be investiaged and a PSI greater than 25% suggests the model should be revaluated on more recent data. PSI can be calculated using the Equation (2.7)(Yurdakul 2018).

$$PSI = \sum_{i=1}^{B}(y_i - y_{b_i}) * ln(\frac{y_i}{y_{b_i}}) \tag{2.7}$$

where y_i is the proportion of target year credit scores that fall in the ith bin, y_{b_i} is the proportion of base year credit scores than fall in the ith bin and B represents the number of bins.

2.4 Data Cleaning Methods

Missing data is a problem that comes with any raw dataset that you might come across and there are a variety of methods to handelling them. Missing data can be categorised into 3 types, missing completely at random, MCAR. Missing at random, MAR, and missing not at random, MNAR. The most common assumption is that the data is MAR, missing data is not missing because of the value but rather a function from some other observered variable. (e.g. an applicant with category A may not want to disclose the value of the variable where as an applicant in category B is more likely to disclose). MCAR is when the probability of missingness is unreleated to the observed variables such as a study participant not returning for a follow-up, etc. MNAR is when the variable is missing due to the value of the variable, for example someone may not want to disclose their income if they consider it to be on the lower end (Buhi, Goodson, and Neilands 2008). Determining what category the variable lies in is often impossible in practice as to determine if a variable is MNAR you would need to know what those missing values are to compare with the values not missing which is often not available (Newman 2014).

For this project I decide to use a mixture of data cleaning methods, regression imputation, single imputation and listwise deletion. Regression imputation is the use of other available data to develop a regression model to predict the missing values. Single imputation is the use of the available data from the variable and impute the same value for each, e.g. the use of mean, median, mode as the value for the remaining. Listwise deletion is the removal of any observation with missing data remaining.

2.5 Chimerge Discretization

For the application of the woe methods a python package called scorecardpy will be used to help automate the process by finding the optimal bins for the numerical variables. The package has the two options for optimizing, tree based and chimerge. For this project I will be using the chimerge method and will explain its application.

The chimerge methods uses the χ^2 statistic to bin numerical variables. It can be seen in detail in (Kerber 1992). The intial step is for the variables to be sorted and then each observation will be split into it's own bin. Each bin will then be compared to its adjacent and calculate the χ^2 value, the bin is then merged into the adjcent bin with the lowest χ^2 value. This step is repeated until all pairs have χ^2 values exceeding a threshold. The formula for computing χ^2 can be seen in Equation (2.8).

$$\chi^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \tag{2.8}$$

where, m = 2, the two intervals being compared. k is the number of classes, in our case 2 (Good and Bad). A_{ij} is the number of observations in ith interval and jth class. E_{ij} is the expected frequency of A_{ij} which is calculated by Equation (2.9).

$$A_{ij} = \frac{R_i - C_j}{N} \tag{2.9}$$

where, R_i is the number of observations in ith interval. C_j is the number of observations in jth class and N is the total number of observations.

Chapter 3

Data

The data I am using for this project is a collection of observations of 5,960 home equity loans which is provided by (Baesens, Roesch, and Scheule 2016). Home equity loans are when an applicant borrows agains the value or 'equity' of their home. You can find a full description of each variable in Table (3.1).

Variable	Definition
BAD	1 = Applicant defaulted on loan or seriously delinquent; 0 = applicant pain load
LOAN	Amount of requested loan
MORTDUE	Amount due on exisiting mortgage
VALUE	Value of property the loan is to go against
REASON	The reason the applicant is applying for the loan. DebtCon = Debt condsolidation; HomeImp = Home Improvement
JOB	Occupational categories
YOJ	Years at present job
DEROG	Number of major derogatory reports
DELINQ	Number of delinquent credit lines
CLAGE	Age of oldest credit line in months
NINQ	Number of recent credit inquiries
CLNO	Number of credit lines
DEBTINC	Debt-to-income ratio

Table 3.1: Variables used in the Data Set

3.1 Data Cleaning

The data provided needed some initial cleaning. 2,596 observations were missing atleast one value with some missing several variables. The biggest culprit of this would be DEBTINC with 1,267 missing values. I decided to handle these missing values on a case by case basis applying different methods. Before I went forward with any imputing I considered any possible outliers within my numerical data, using the summary Table B.1. You can see for the quantile ranges that there will most likely be some outliers occuring in the majority of the numerical variables. The gerneral consesus is that there is a necessity to handle outliers but a lack of definitive way to handle them (Nyitrai and Virág 2019). For this project I chose to removed the 99th percentile for every numerical variable excluding BAD, this ended up removing 613 rows.

Moving onto imputing variable, for MORTDUE and VALUE I imputed their values using a simple linear regression of the other. This was going on the assumption that the mortgage due on a house had a strong relationship with the value of property. The assumption is further backed up with the correlation between the two being 0.8748 before imputing, far higher than any of the other correlations in the data. So for MORTDUE I used Equation (3.1) and for VALUE I used Equation (3.2). This was applied to any missing value where the other was present and for the remaining I took the mean of each variable from the original data before the imputations.

$$\text{MORTDUE} = \beta_0 + \beta_1 \text{VALUE} \tag{3.1}$$

β_0	-2145.6497
β_1	0.7177

$$\text{VALUE} = \beta_0 + \beta_1 \text{MORTDUE} \tag{3.2}$$

β_0	21340.4803
β_1	1.1253

For the remaining numerical variables excluding DEBTINC I chose to take the median of the values as there were only a small amount missing from each but still a significant amount ($> 4\%$) missing. There is an argument that because DEBTINC is missing 1,081 (20.2 %) that some other method from using the median value should be used. After some further analysis the decsion to drop the variable was made, imputing did not appear to be an option as of the 915 bad applicants, 634 (69.3%) of them were missing DEBTINC compared to 4432 and 447 (10.1%) for good applicants. Dropping every row with DEBTINC did not appear to be practical either as it would result in the loss of almost 70% of the bad applicants and their data whilst also taking the bad rate down to 6%. Dropping DEBTINC would be the lower loss of information against the alternative of dropping missing rows (5,347 values lost versus 12,972).

Last was the two categorical variables REASON (DebtCon, HomeImp) and JOB (Other, Office, Sales, Mgr, ProfExe, Self). REASON's categories were specified in the data dictionary but JOB's categories were not. The decision was made to use listwise deletion on these as they were the only variables remaining with missing data. This resulted in the removal of 377 rows.

With these two completed I had no more missing values and no other noticeable issues which needed to be corrected before I could further look into the variables. A summary of actions taken on missing values can be found in Table (3.2).

Variable	No. Missing	Solution
BAD	0	N/A
LOAN	0	N/A
MORTDUE	477	Imputed from a linear regression (3.1). Mean taken when VALUE was unavailable
VALUE	86	Imputed from a linear regression (3.2). Mean taken when MORTDUE was unavailable
REASON	222	Listwise deletion
JOB	258	Listwise deletion
YOJ	486	Median taken
DEROG	655	Median taken
DELINQ	548	Median taken
CLAGE	303	Median taken
NINQ	476	Median taken
CLNO	222	Median taken
DEBTINC	1081	Variable dropped as too much information missing

Table 3.2: Missing Variables Breakdown

3.2 Variables

With data cleaned the remaining observations was 4970 of 12 independ variables with no missing values. A summary of the numerical values can be found in Table (B.2). There you can see the remaining data has a bad rate of 18% which equates to 876 defaulted applicants.

LOAN

LOAN, the amount requested for the home equity loan by the applicant can be seen in Figure (3.1). The initial assumption was that higher loan values would have a higher bad rate due to the larger amount to pay back, increasing the length and difficulty for the applicant to pay back the loan. Looking at the figure this does not appear to be the case, although small, larger loans tend to be payed off more often. The reasons for this are unclear, a couple suggested could be that since we do not know exactly how this data was gathered the case could be that for larger loans the bank/company offering these loans had higher cutoffs on their applicant scoring to prevent higher risks in higher potential loss causing a shift down in the bad rate. Another could be the arguement that larger requested loans are coming from owners of higher valued properties, which could be the case when looking at the correlation Table (B.3). A higher property value indicates a better economic status and less likely to default on a loan. It would be reasonable to expect this variable to have a significant effect on the credit score.

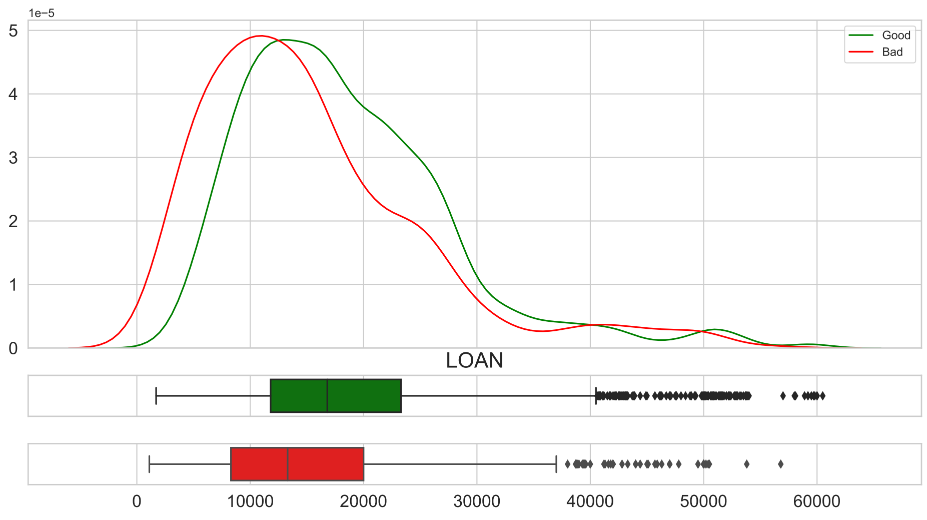

Figure 3.1: Distribution of LOAN by BAD.

MORTDUE

MORTDUE, the outstanding balance on the applicants existing mortgage. Assumption here would be similar to LOAN, a higher outstanding balance on their mortgage mean a large amount of debt and an increased risk of defaulting due to the larger payments. Looking at Figure (3.2) you can see it does not follow this assumption, again a small but clear difference in the distribution shows that applicants with a higher outstanding balance on their mortgage are less likely to default. Whatever the reason behind this is would most likely be the same as the reason behind LOAN.

VALUE

VALUE, the property of the applicants and the equity the loan is being put against. The same initial assumption being made with MORTDUE is also here, a value of an applicants property is an indication of their economic status. An owner of a higher valued property should be able to payback a larger loan and less likely to default on smaller ones. From Figure (3.3) there is a small visible effect of loan on their probability of defaulting.

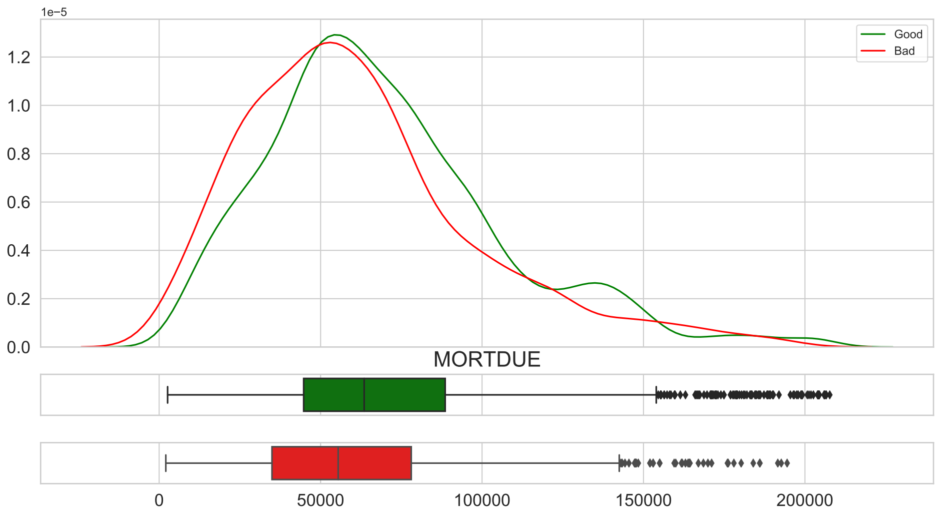

Figure 3.2: Distribution of MORTDUE by BAD.

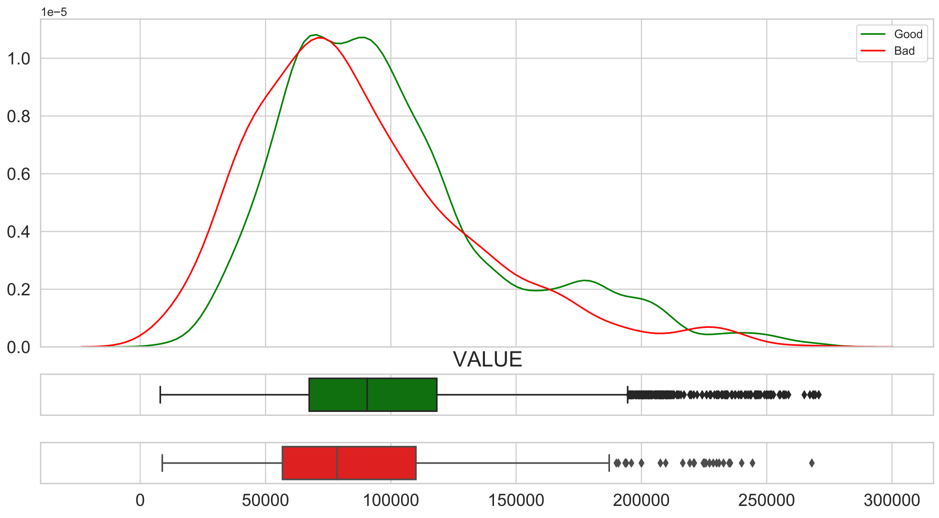

Figure 3.3: Distribution of VALUE by BAD.

REASON

REASON, the reason for the applicant's request. There are only two categories as seen in 3.4, DebtCon, the loan would be used for a debt consolidation. HomeImp, the loan is being used for a home improvement. A breakdown of their splits between good and bad can be found in Table (3.3). The two categories continue to have similar splits when they do not default but when they do default, HomeImp's split increases by 7%. Looking at this further in Figure (3.4) the bad rate for DebtCon is 16.24% and HomeImp 20.75%. Although there does appear to be a difference, it is small and compared to other variables in the group I would expect this to be on the lower end of signifiance.

Category	% of Total (N = 4970)	% of Good (N = 4094)	% of Bad (N = 876)
DebtCon	69.3% (3442)	70.4% (2883)	63.8% (559)
HomeImp	30.7% (1528)	29.6% (1211)	36.2% (317)

Table 3.3: REASON breakdown

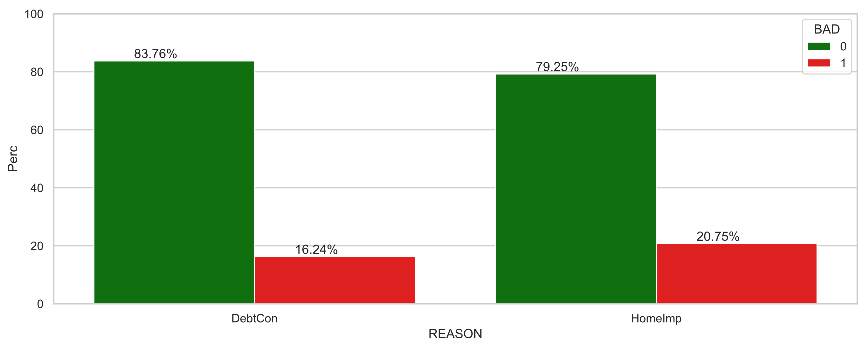

Figure 3.4: Category plot of REASON by BAD.

JOB

JOB, categorical job occupation. Categories can be seen in 3.5. Occupation could be used as an indicator for the applicants economic status e.g. a ProfExe, proffesional executive is more likely to have a higher income than office staff or someone who is self employed. It could also be used to see how volatile their employement status is, someone who is self employed can be seen as a possible risk due to their income being potentially unstable. Although we do not have a way of looking at their job security, we do have YOJ, years at present job, as an indicator of Job security. Comparing Figure (3.6) and Figure (3.5), Sales, the job with the lowest average YOJ has the highest bad rate at 31.52% followed by Self at 24.14%.

Category	% of Total (N = 4970)	% of Good (N = 4094)	% of Bad (N = 876)
Other	42.5% (2110)	40.7% (1666)	50.7% (444)
ProfExe	22.5% (1119)	23.9% (977)	16.2% (142)
Office	17.0% (844)	18.2% (746)	11.2% (98)
Mgr	13.3% (660)	13.0% (532)	14.6% (128)
Self	2.9% (145)	2.7% (110)	4.0% (35)
Sales	1.9% (92)	1.5% (63)	3.3% (29)

Table 3.4: JOB breakdown

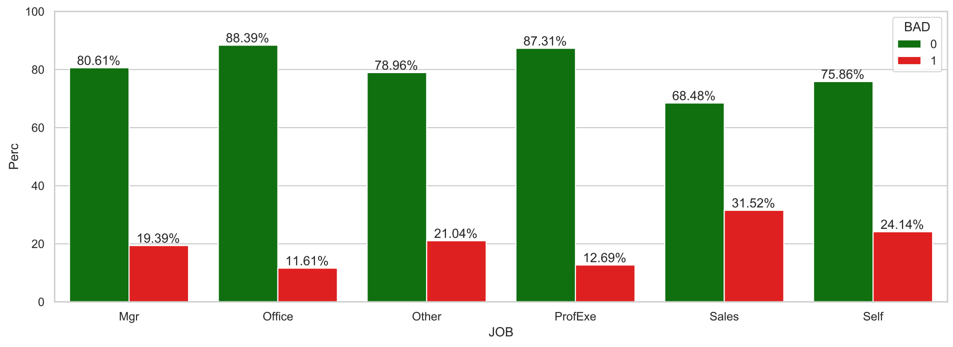

Figure 3.5: Category plot of JOB by BAD.

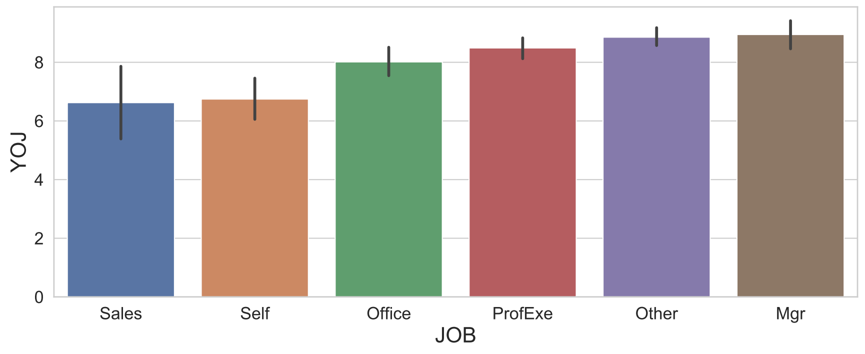

Figure 3.6: Category plot of JOB by mean of YOJ.

YOJ

YOJ, number of years the applicant has been at present job. Can be an indicator of job security, an applicant losing their job can be a high risk of defaulting on their loan. Figure (3.7 shows that the majority of applicants are between 3 to 14 years at their current job with the mean for bads and goods being relatively the same. The difference between them starts to become noticable at higher values where an applicant who has been at their present job for more than 20 years starts to become less likely to default.

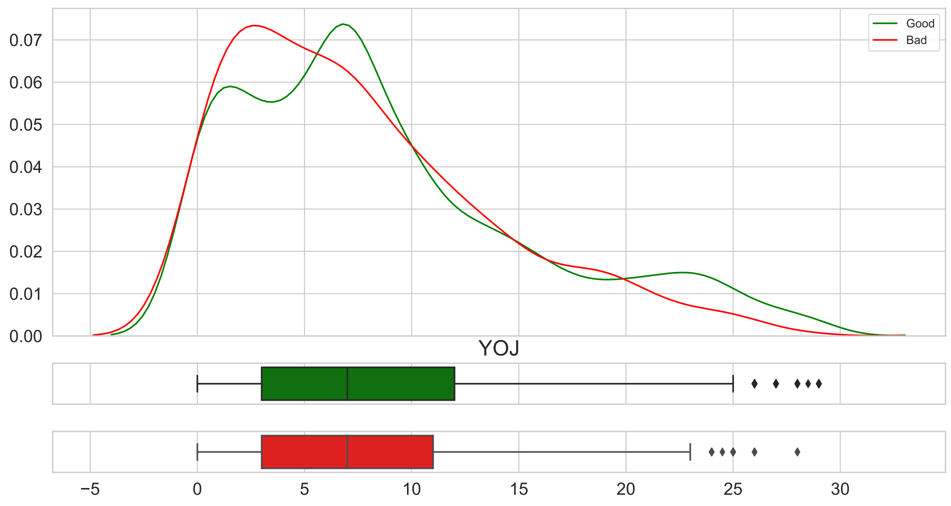

Figure 3.7: Distribution of YOJ by BAD.

DEROG

DEROG, number of dergoatory marks on aginast the applicant. A derogatory mark can have a large impact on your chances of being accepted for a loan. In this dataset, only 12.3% of applicants have 1 or more dergoatory mark. An applicant can recieve a derogatory mark for various reasons such as, missing payments, bankruptcy, repossession, etc. The severity of the reason for the derogatory mark is often used in credit scoring but for this dataset we only have the numbers of marks against the applicant. Figure (3.8) shows the impact having a derogatory mark can have on your chance of defaulting with 0 having a bad rate of 15.04% and 3 having a bad rate of 65.12%.

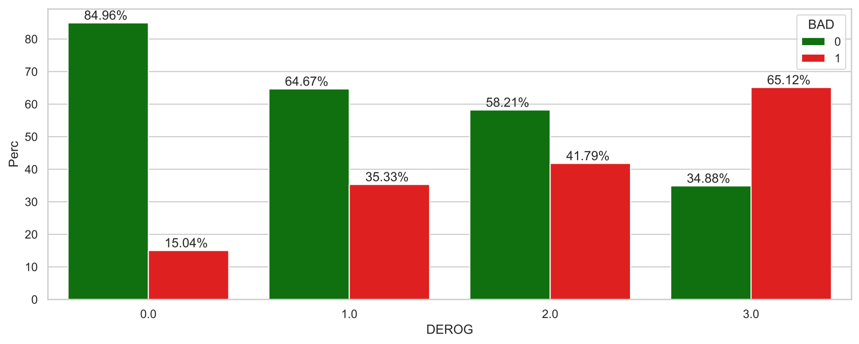

Figure 3.8: Category plot of DEROG by BAD.

DELINQ

DELINQ, Number of delinquent credit lines. Delinquency is used to describe when a borrower has missed a payment, the borrower is referred to as delinquent. Delinquency preceeds defaulting, once a borrower has been deqlinquent for a time it comes apparent that the borrower is unable to pay back the loan. The data we are provided is the number of deqlinquent credit lines, which is the number of credit lines the borrower has missed payments on.

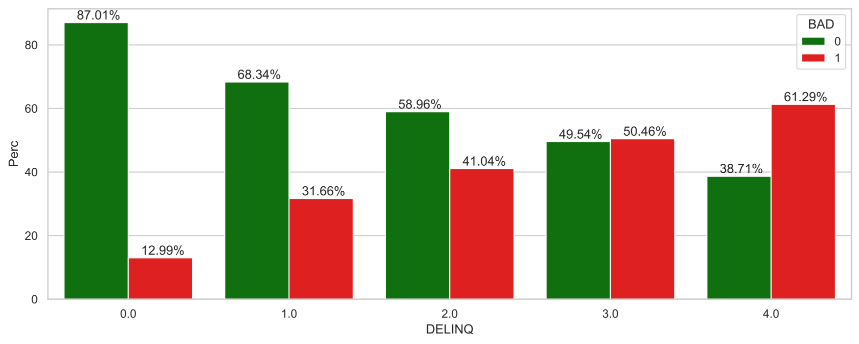

Figure 3.9: Category plot of DELINQ by BAD.

CLAGE

CLAGE, Age of oldest credit line in months. Assumption here is the longer an applicant has held credit lines the more experienced they are in repaying payments. A applicant who has experience in credit repayments for 20 years is going to have a better time in avoiding delinqency and defaulting that someone who is new to credit. Figure (3.10) reinforces this assumption, there is a clear difference in the distributions of goods and bads. Applicants between 100 and 150 months on thier oldest credit line have a bad rate of 21.8% and applicants between 200 and 300 have a bad rate of 9.5%. Expecting this value to have a strong signifance for the credit score.

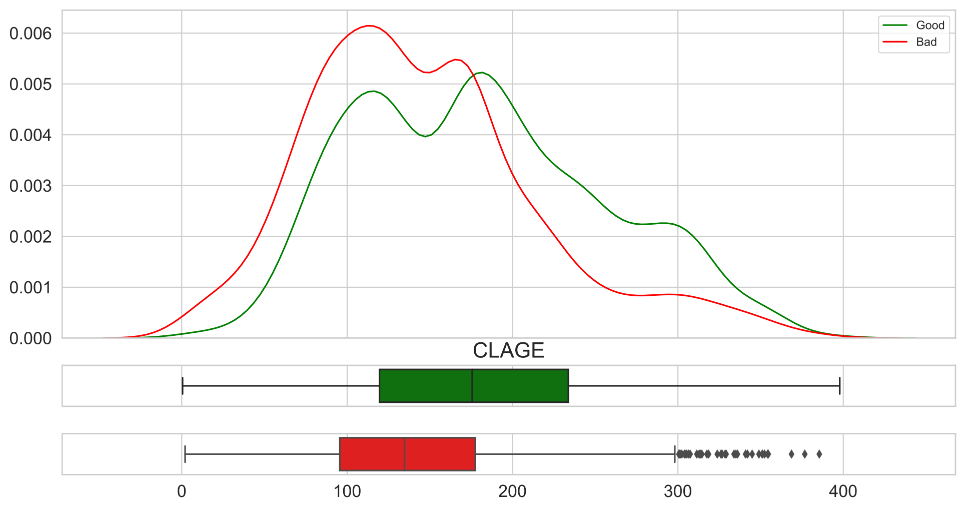

Figure 3.10: Distribution of CLAGE by BAD.

NINQ

NINQ, number of recent credit inquiries. Figure (3.11) indicates that the more recent credit equiries an applicant makes the more likely they are to default, this trend is followed until the higher values are reached but this deviation could be due to the low sample size at larger values. 73.8% of applicants have made no more than 1 recent credit equiry where as only 3% of applicants made 5 or more. This could create an issue depending on how the WOE method bins the values.

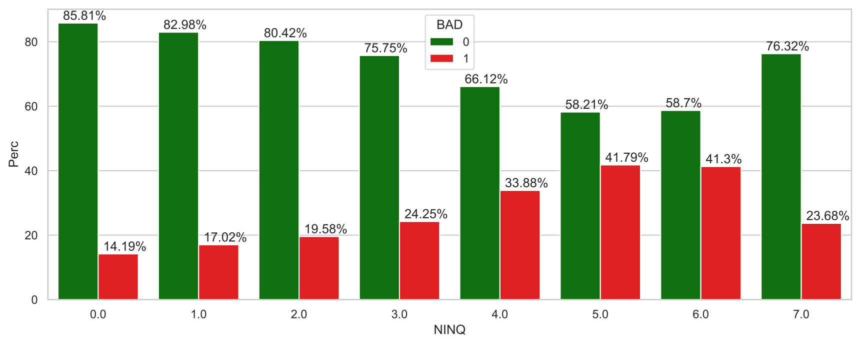

Figure 3.11: Category plot of NINQ by BAD.

CLNO

CLNO, number of credit lines. A credit line can be any method in which someone can recieve credit such as an overdraft, credit card, etc. Figure (3.12) suggests that applicants with low or high values for credit lines have a higher bad rate that ones in the centre. A reason behind this could be that a applicant with a low number of credit lines could be inexperience with debt management or don't have access to other credit to ensure payment on loans is on time. On the other end it could be that applicants with a large number of credit lines become incapable of managing the potential debt from the numerous sources. Where as the centre is seen to be where applicants have a good control over their credit lines. Applicants with a CLNO value between 20 and 30 have a bad rate of 15.1% where applicants outside of this group have a bad rate of 23.1%. A differnce of 8%.

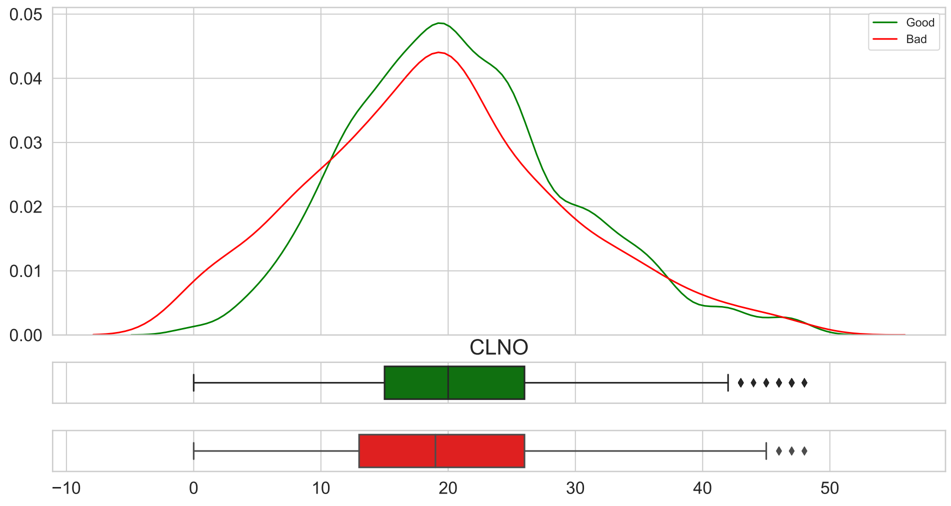

Figure 3.12: Distribution of CLNO by BAD.

3.3 WOE and IV

WOE binning and calculation was done using the scorecardpy package provided by ShichenXie (*Python Package for Credit Scoring*), results can be found in the two Tables (3.15) & (3.16). I allowed the function to determine optimal bins itself using the chimerge method described in Section 2.5. In the table we have a breakdown for each bin providing their woe value and individual information values. We can also see the total information value of each variable in the last column (total_iv). Based on IV, our strongest predictor by a large margin is DELINQ with a IV value of 0.40 and our weakest is Reason with an IV of 0.02. Referring back to Bailey's guideline (2.1)(Bailey 2004), we can categorise each variable on their strength from the IV.

Looking back on our data exploration we can see that our assumption about some of the variables appear to be reflecting in the WOE bins. Reason, a variable I expected to be rather insignifcant due to the bad rate being similar in both categories has the lowest IV. NINQ appears to have been binned appropriately with the upper bin being [4.0, inf) meaning the drop in bad rate at 7.0, assumed to be from a small sample, should not have a siginicant effect on the woe value of the group. The assumption on CLAGE, the variable with the second highest IV, is also reflected in the WOE values with a clear trend appearing in the bad probabilities of each bin seen in Figure (3.14). The increase in bad probability in bin [170.0, 180.0) is most likely an error created by the single imputation used in the data cleaning, referring to the summary table before cleaning, Table (B.1), you can see the median is 173.63 which is within the bin not following the trend.

Variables Prediction Strength

Variable	Strength	IV
LOAN	Average	0.28
MORTDUE	Weak	0.08
VALUE	Average	0.18
REASON	Poor	0.02
JOB	Average	0.11
YOJ	Weak	0.08
DEROG	Average	0.21
DELINQ	Strong	0.40
CLAGE	Average	0.29
NINQ	Average	0.10
CLNO	Average	0.15

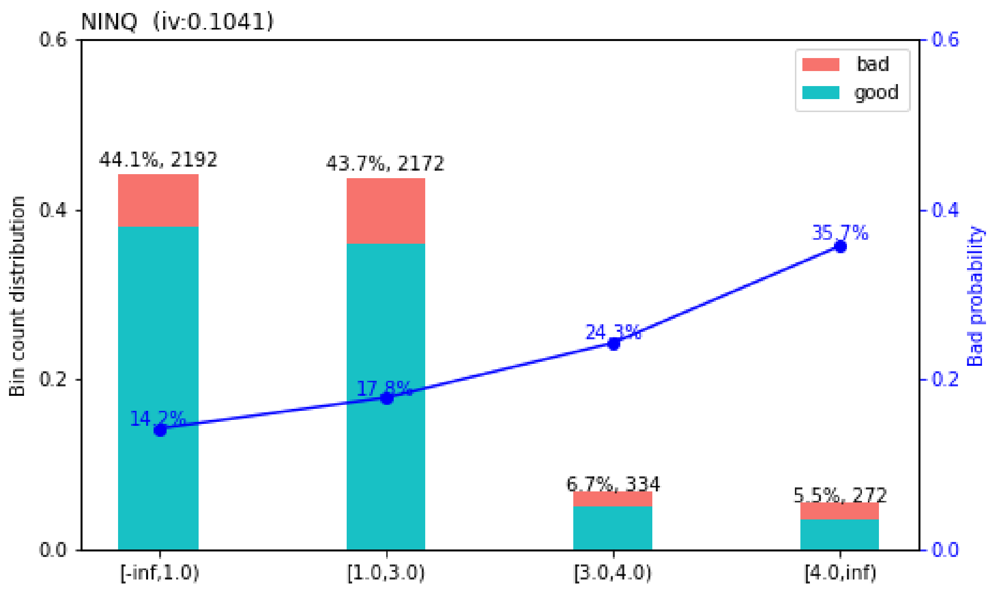

Figure 3.13: NINQ woe plot

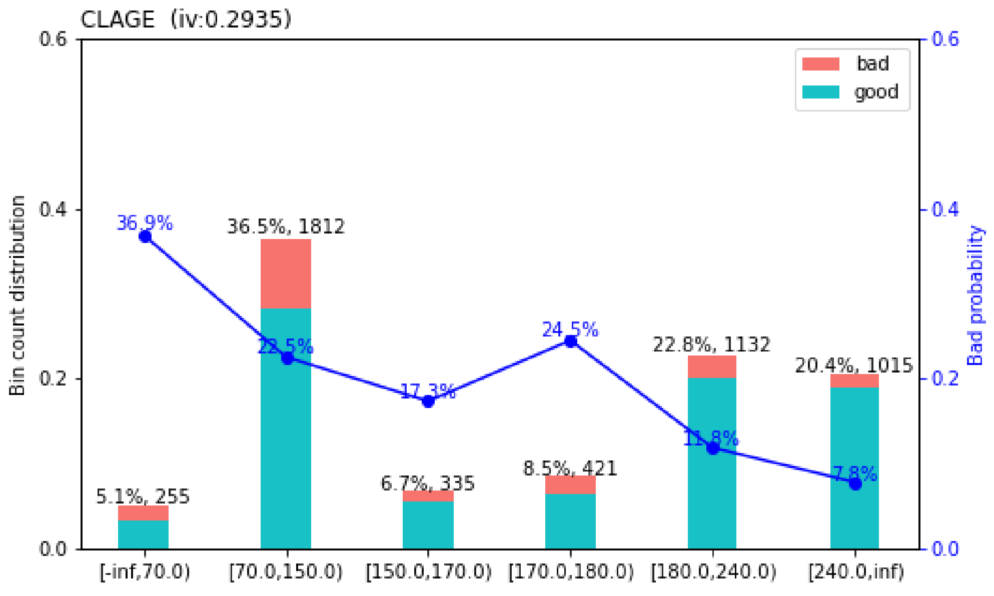

Figure 3.14: Clage woe plot

variable	bin	count	count_distr	good	bad	badprob	woe	bin_iv	total_iv
LOAN	[-inf, 6000.0)	250	0.05	125	125	0.50	1.54	0.17	0.28
	[6000.0, 8000.0)	276	0.06	202	74	0.27	0.54	0.02	0.28
	[8000.0, 10000.0)	428	0.09	352	76	0.18	0.01	0.00	0.28
	[10000.0, 11000.0)	259	0.05	199	60	0.23	0.34	0.01	0.28
	[11000.0, 15000.0)	927	0.19	778	149	0.16	-0.11	0.00	0.28
	[15000.0, 16000.0)	296	0.06	226	70	0.24	0.37	0.01	0.28
	[16000.0, inf)	2534	0.51	2212	322	0.13	-0.39	0.07	0.28
MORTDUE	[-inf, 35000.0)	874	0.18	653	221	0.25	0.46	0.04	0.08
	[35000.0, 55000.0)	1139	0.23	928	211	0.19	0.06	0.00	0.08
	[55000.0, 60000.0)	341	0.07	293	48	0.14	-0.27	0.00	0.08
	[60000.0, 75000.0)	854	0.17	697	157	0.18	0.05	0.00	0.08
	[75000.0, inf)	1762	0.35	1523	239	0.14	-0.31	0.03	0.08
VALUE	[-inf, 50000.0)	521	0.10	347	174	0.33	0.85	0.10	0.18
	[50000.0, 70000.0)	982	0.20	815	167	0.17	-0.04	0.00	0.18
	[70000.0, 80000.0)	508	0.10	401	107	0.21	0.22	0.01	0.18
	[80000.0, 125000.0)	1922	0.39	1657	265	0.14	-0.29	0.03	0.18
	[125000.0, 175000.0)	608	0.12	484	124	0.20	0.18	0.00	0.18
	[175000.0, inf)	429	0.09	390	39	0.09	-0.76	0.04	0.18
REASON	DebtCon	3442	0.69	2883	559	0.16	-0.1	0.01	0.02
	HomeImp	1528	0.31	1211	317	0.21	0.2	0.01	0.02
JOB	Other	2110	0.42	1666	444	0.21	0.22	0.02	0.11
	Office	844	0.17	746	98	0.12	-0.49	0.03	0.11
	Sales	92	0.02	63	29	0.32	0.77	0.01	0.11
	ProfExe	1119	0.23	977	142	0.13	-0.39	0.03	0.11
	Mgr	660	0.13	532	128	0.19	0.12	0.00	0.11
	Self	145	0.03	110	35	0.24	0.40	0.01	0.11

Figure 3.15: WOE results table.

variable	bin	count	count_distr	good	bad	badprob	woe	bin_iv	total_iv
YOJ	[-inf, 2.0)	751	0.15	616	135	0.18	0.02	0.00	0.08
	[2.0, 6.0)	1139	0.23	880	259	0.23	0.32	0.03	0.08
	[6.0, 10.0)	1356	0.27	1155	201	0.15	-0.21	0.01	0.08
	[10.0, 23.0)	1421	0.29	1162	259	0.18	0.04	0.00	0.08
	[23.0, inf)	303	0.06	281	22	0.07	-1.01	0.04	0.08
DEROG	[-inf, 1.0)	4442	0.89	3774	668	0.15	-0.19	0.03	0.21
	[1.0, inf)	528	0.11	320	208	0.39	1.11	0.18	0.21
DELINQ	[-inf, 1.0)	4050	0.81	3524	526	0.13	-0.36	0.09	0.4
	[1.0, 2.0)	537	0.11	367	170	0.32	0.77	0.08	0.4
	[2.0, inf)	383	0.08	203	180	0.47	1.42	0.22	0.4
CLAGE	[-inf, 70.0)	255	0.05	161	94	0.37	1.00	0.07	0.29
	[70.0, 150.0)	1812	0.36	1404	408	0.23	0.31	0.04	0.29
	[150.0, 170.0)	335	0.07	277	58	0.17	-0.02	0.00	0.29
	[170.0, 180.0)	421	0.08	318	103	0.24	0.41	0.02	0.29
	[180.0, 240.0)	1132	0.23	998	134	0.12	-0.47	0.04	0.29
	[240.0, inf)	1015	0.20	936	79	0.08	-0.93	0.13	0.29
NINQ	[-inf, 1.0)	2192	0.44	1881	311	0.14	-0.26	0.03	0.1
	[1.0, 3.0)	2172	0.44	1785	387	0.18	0.01	0.00	0.1
	[3.0, 4.0)	334	0.07	253	81	0.24	0.40	0.01	0.1
	[4.0, inf)	272	0.05	175	97	0.36	0.95	0.06	0.1
CLNO	[-inf, 10.0)	479	0.10	323	156	0.33	0.81	0.08	0.15
	[10.0, 20.0)	1882	0.38	1589	293	0.16	-0.15	0.01	0.15
	[20.0, 21.0)	297	0.06	221	76	0.26	0.47	0.02	0.15
	[21.0, 24.0)	600	0.12	505	95	0.16	-0.13	0.00	0.15
	[24.0, 26.0)	403	0.08	369	34	0.08	-0.84	0.04	0.15
	[26.0, inf)	1309	0.26	1087	222	0.17	-0.05	0.00	0.15

Figure 3.16: WOE results table.

Chapter 4

Modelling

4.1 Results

For the modelling, the data was split into train and test sets with proportions 0.7 and 0.3 respectively. The bad rate was maintained in each data at 17% to ensure fairness and the variables were converted to their woe values for their respective bin from tables (3.15) & (3.16). Once split the train data was passed through a glm model with logit link, for this the python package statsmodels was used *Python Package for Statistical Modelling*. Three models were created, first, seen in Table (4.1), was the base model with every variable included with no changes. Looking at the results table we can see that the majority of variables are highly significant with p-values less than 0.001. Mortdue is deemed the most insignificant, most likely due to its high correlation with VALUE and as such the variance it explains is already captured by VALUE. This is confirmed if we run the model again with VALUE dropped, seen in Table (B.1). The p-value for MORTDUE is now 0.0005.

For the second model I looked at applying log transformations to the variables which had a heavy right skew, LOAN, MORTDUE, VALUE and YOJ to try to improve their p-values. These values were then passed through the woe binning and the values converted. The IV of LOAN and YOJ improved by 0.01 and 0.03 respectively whilst MORTDUE and VALUE's IV decreased, based on this only the log transformations of LOAN and YOJ were kept and used for the second model. The results for the second model can be seen in Table (B.2).

Finally, a third model was made from dropping variables with high p-values, these were MORTDUE and REASON. The results are displayed in Table (4.2). All the remaining variables have a p-value of less than 0.0001. Every coefficient is positive, implying that an increase in all variables increase in the probability of BAD, which from initial observation shouldn't be right. Looking back on Section (3.2) we have some assumptions of large values decreasing the probability of BAD such as CLAGE and YOJ. The reason for this is a result of the WOE methods, changing the values of our binned variables to their respective WOE value seen in Tables (3.15) & (3.16). Higher values of CLAGE have negative values, for example values of CLAGE between 180 and 240 have a woe value of -0.47. So in reality, a CLAGE value in this range would have the effect of decreasing the probability of BAD.

The performance of each model is shown and compared in Table (4.3). It can be seen from this table that based on the performance evaulation mentioned in Section 2.3. Model 2 and 3

Model:	GLM	AIC:	2514.3936
Link Function:	logit	BIC:	-25781.2585
Dependent Variable:	BAD	Log-Likelihood:	-1245.2

	Coef.	Std.Err.	z	P> |z|	[0.025	0.975]
const	-1.5770	0.0539	-29.2635	0.0000	-1.6826	-1.4714
DELINQ_woe	1.0403	0.0790	13.1746	0.0000	0.8855	1.1951
CLAGE_woe	0.9281	0.1005	9.2313	0.0000	0.7311	1.1252
NINQ_woe	1.0073	0.1533	6.5722	0.0000	0.7069	1.3077
MORTDUE_woe	-0.1110	0.2344	-0.4737	0.6357	-0.5705	0.3484
CLNO_woe	0.8184	0.1343	6.0959	0.0000	0.5553	1.0816
LOAN_woe	0.8540	0.1018	8.3890	0.0000	0.6545	1.0535
REASON_woe	-0.4899	0.3985	-1.2295	0.2189	-1.2709	0.2910
JOB_woe	0.8730	0.1629	5.3592	0.0000	0.5537	1.1923
VALUE_woe	0.8503	0.1535	5.5400	0.0000	0.5495	1.1512
DEROG_woe	0.7714	0.1046	7.3725	0.0000	0.5663	0.9764
YOJ_woe	0.8313	0.1992	4.1740	0.0000	0.4410	1.2217

Table 4.1: Results: Model 1

Model:	GLM	AIC:	2494.3355
Link Function:	logit	BIC:	-25813.6256
Dependent Variable:	BAD	Log-Likelihood:	-1237.2

	Coef.	Std.Err.	z	P> |z|	[0.025	0.975]
const	-1.5749	0.0540	-29.1591	0.0000	-1.6808	-1.4691
DELINQ_woe	1.0369	0.0791	13.1031	0.0000	0.8818	1.1920
CLAGE_woe	0.9499	0.1007	9.4281	0.0000	0.7524	1.1473
NINQ_woe	1.0102	0.1525	6.6229	0.0000	0.7112	1.3091
CLNO_woe	0.7860	0.1331	5.9031	0.0000	0.5250	1.0469
LOAN_woe	0.8042	0.0934	8.6116	0.0000	0.6212	0.9872
JOB_woe	0.8508	0.1630	5.2195	0.0000	0.5313	1.1703
VALUE_woe	0.7937	0.1230	6.4517	0.0000	0.5526	1.0348
DEROG_woe	0.7840	0.1048	7.4802	0.0000	0.5785	0.9894
YOJ_woe	0.8988	0.1660	5.4148	0.0000	0.5735	1.2241

Table 4.2: Results: Model 3

Model	AIC	KS	GINI	Divergence
Model 1 (Default)	2514.39	0.4581	0.5956	1.3802
Model 2 (Log transformations)	2495.42	0.4814	0.6078	1.4439
Model 3 (REASON and MORTDUE dropped)	2494.34	0.4822	0.6116	1.4568

Table 4.3: Performance Evaluation Results On Test

appear to out perform Model 1 but when comparing the two the difference between performance indicators becomes smaller. In the case of Model 3 the AIC is lower but the GINI coefficient is also lower than Model 2. Based on Table (4.3) either Model 2 or 3 would be an appropriate choice but I decided to go with Model 3. This is because the difference in AIC is only 1.08 and when looking at the Log-Likelihood for each model the difference is 1.5. This is implying that the performance based on AIC is only better because of the removal of two variables making it a smaller and less complicated model. Where as the GINI coefficient doesn't consider the complication of the model.

In Figure (4.1) the results from the logisitc regression have been used and converted into a scorecard. The figure displays the distribution and bad probability for the test and train data sets. You can see the distributions appear to follow the same trend but on the lower end seem to vary slightly, this is most likely due to the smaller sample size of clients that have been scored in this range. The PSI value is 1.1% indicating the two samples don't have a shift in population. In Figures (4.2) & (4.3) we can see a clear seperation of goods and bads within the scorecard with bads clearly on the lower end of the scorecard and goods on the upper.

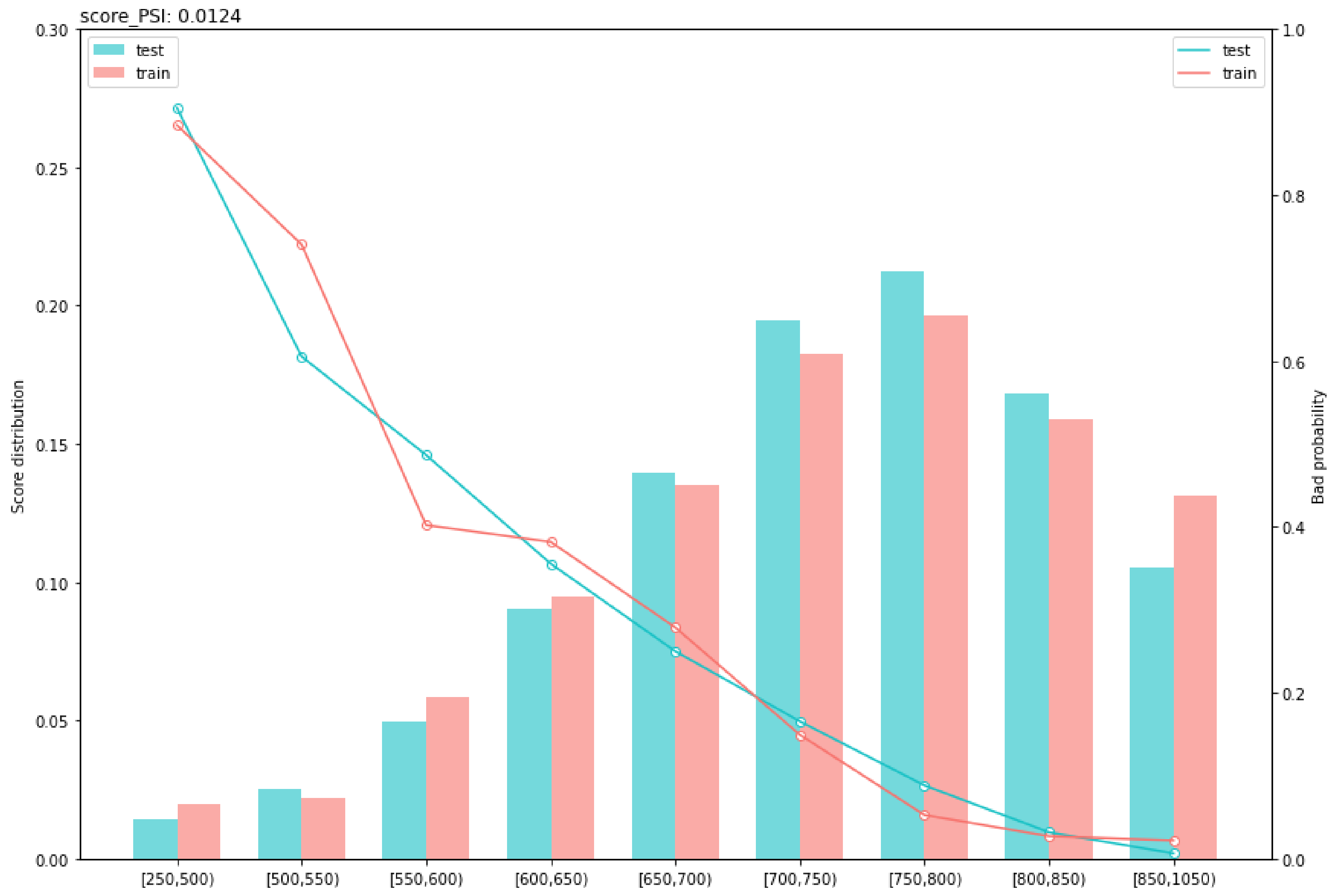

Figure 4.1: Scorecard Plot

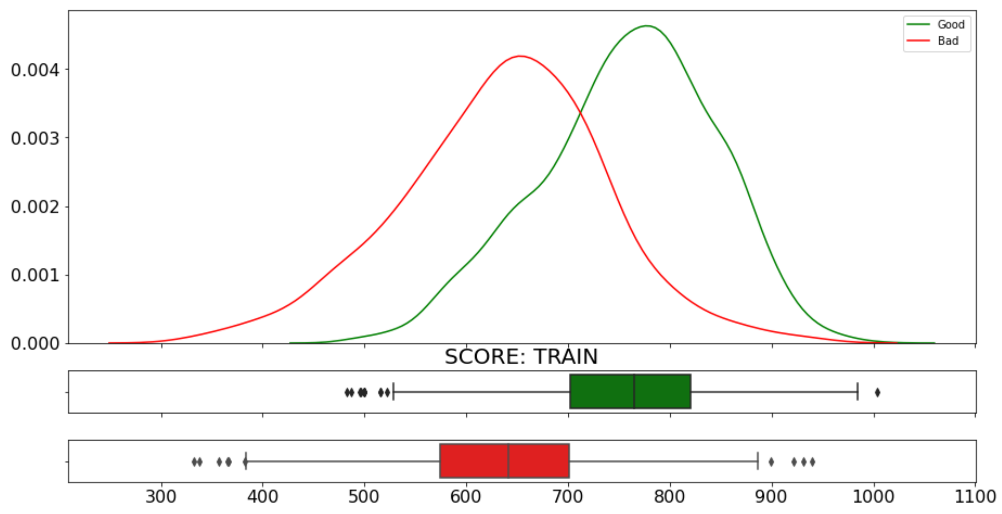

Figure 4.2: Scorecard Distribution for Train data

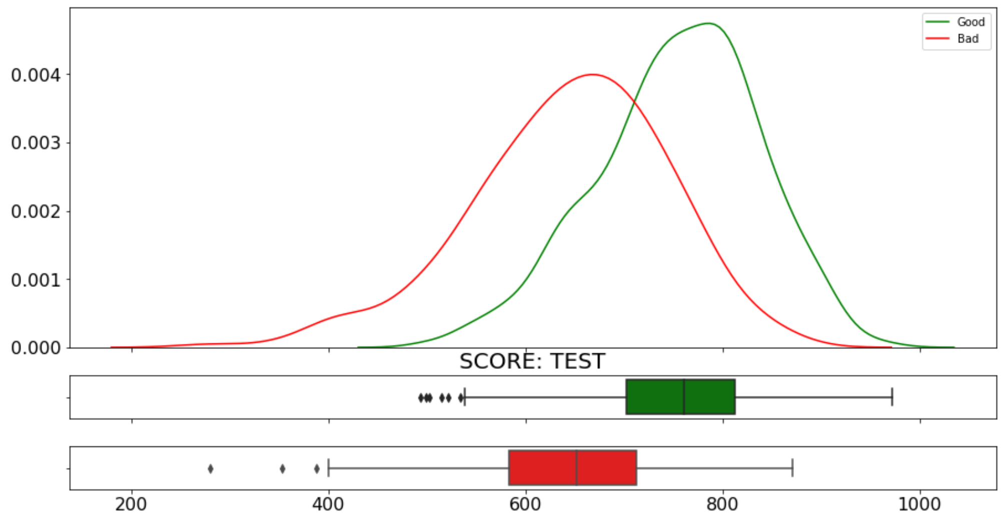

Figure 4.3: Scorecard Distribution for Test data

Chapter 5

Alternative Uses: Covid-19

Scorecard modelling techniques have proven to be effective methods in other areas of science such as medicine. One such present use is to develop a scorecard of the health risk of patients with Covid-19. Many such papers have been published on the potential risk of patients with various characteristics such as age, sex, race, underlying health conditions. Using an live open data set on Covid-19 patients (Group 2020), I attempted to apply the credit scoring methods to develop a scorecard for a patients health risk.

The data set at time of downloading contained 2,676,311 rows of patient data. The majority of the data set was limited containing only, an ID, country of patient and date of contracting, but some rows contained more information on the patients such as age, sex, symptoms, chronic diseases. I subsetted my data down to rows which contained, age, sex and symptoms and made the assumption that any of these rows which had a null value for chronic disease indicates no disease present. This left me with 215 rows.

Variable	Definition
outcome	1 = Died; 0 = Survived
age	Age of patient
sex	Sex of patient (1 = Male) (0 = Female)
cough	Wether the patient had a cough
fever	Wether the patient had a fever
pneumonia	Wether the patient had pneumonia
respiratory problems	Wether the patient was having repiratory problems, could range from acute symptoms to respiratory failure
hypertension	Wether the patient has a chronic disease of hypertension

Table 5.1: Variables used in the Data Set

	outcome	age	sex	cough	fever	pneumonia	respiratory problems	hypertension
count	202.00	202.00	202.00	202.00	202.00	202.00	202.00	202.00
mean	0.68	59.95	0.68	0.33	0.44	0.34	0.30	0.36
std	0.47	18.38	0.47	0.47	0.50	0.47	0.46	0.48
min	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00
25%	0.00	46.00	0.00	0.00	0.00	0.00	0.00	0.00
50%	1.00	63.50	1.00	0.00	0.00	0.00	0.00	0.00
75%	1.00	73.75	1.00	1.00	1.00	1.00	1.00	1.00
max	1.00	89.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 5.2: Data summary

Next I split symptoms and diseases into their own dummy variables and removed any with few observations ($< 20\%$). The final dataset can be seen in Table (5.1). It should be noted that diabetes was included in the list of chronic diseases but because every patient with diabetes in the data died it was creating issues when using logistic regression so it was removed. From Table (5.2) we can see the bad rate is 68% most likely because the patients in this data set have already been admitted to hospital because of Covid-19. The details on other variables can also be found in this table.

After cleaning the data I applied the same methods I used for the credit scoring data set, first woe binning the variables using the scorecardpy package and then applying a logisitic regression to the data. Results of these two can be found in Tables (5.3) & (5.4) respectively. You can see in Table (5.3) that age is the variable with the highest IV at 2.93 and sex has an IV at 0.01.

Cough and fever, although their IV's are high the reason for that appears to be reversed. Both of their bad probabilities are much higher when the symptom is not present, 0.74 and 0.81. My assumption would be that any symptom would increase the probability of death but in these cases it is the opposite. Since we cannot determine how the data was collected for these patients we can only assume possibilites for the reasons. One such reason could be that patients with more severe symptoms such as respiratory problems did not have the symptoms of cough and fever recorded as they were deemed insignificant symptoms compared the prior. This would create a case where patients who only ever developed minor symptoms would be the only ones to have those symptoms recorded.

variable	bin	count	count_distr	good	bad	badprob	woe	bin_iv	total_iv
age	[-inf,36.0)	30	0.14	29	1	0.03	-3.95	1.46	2.93
	[36.0,54.0)	48	0.22	30	18	0.38	-1.09	0.28	2.93
	[54.0,68.0)	61	0.28	14	47	0.77	0.63	0.10	2.93
	[68.0,inf)	76	0.35	4	72	0.95	2.31	1.08	2.93
sex	[-inf,1.0)	68	0.32	27	41	0.60	-0.17	0.01	0.01
	[1.0,inf)	147	0.68	50	97	0.66	0.08	0.00	0.01
cough	[-inf,1.0)	149	0.69	38	111	0.74	0.49	0.15	0.45
	[1.0,inf)	66	0.31	39	27	0.41	-0.95	0.30	0.45
fever	[-inf,1.0)	127	0.59	24	103	0.81	0.87	0.38	0.81
	[1.0,inf)	88	0.41	53	35	0.40	-1.00	0.43	0.81
pneumonia	[-inf,1.0)	147	0.68	75	72	0.49	-0.62	0.28	1.6
	[1.0,inf)	68	0.32	2	66	0.97	2.91	1.32	1.6
respiratory	[-inf,1.0)	155	0.72	75	80	0.52	-0.52	0.2	1.3
	[1.0,inf)	60	0.28	2	58	0.97	2.78	1.1	1.3
hypertension	[-inf,1.0)	142	0.66	75	67	0.47	-0.70	0.34	1.8
	[1.0,inf)	73	0.34	2	71	0.97	2.99	1.46	1.8

Table 5.3: WOE results table.

Now looking at the results of the Logistic Regression, Table (5.4). As expected from the p-values, age is our most siginificant variable with respiratory coming second. Sex, even though it is not significant, it still has the highest coefficient. When looking back to Table (5.3), we see that the bad probability from sex is quite similar, the closest of any other variables, and as such I was not expecting the coefficient to be this large. That along with its insignificance, I chose to drop the variable and compare the resulting model which can be seen in Table (5.5). Both AIC and BIC are lower and the siginificance of the remaining variables stayed relatively the same.

Going one step further I chose to also drop cough as its p-value was 0.8145, much higher than the other variables and then repeated this step until I could no longer improve the AIC. The resulting Table can be seen in (5.6) with cough and hypertension removed. The remaining variables have fairly low p-values with all being less than 0.1 and age remains our most significant variable. Fever remains in the model with the effect of lowering the probability if the symptom is observed, although it is significant for the data I would suspect this would not be reflected in other data sets.

Model:	GLM	AIC:	111.0205
Link Function:	logit	BIC:	-622.4505

	Coef.	Std.Err.	z	P> |z|	[0.025	0.975]
const	0.6338	0.2858	2.2178	0.0266	0.0737	1.1939
cough_woe	0.1159	0.3897	0.2975	0.7661	-0.6478	0.8797
age_woe	0.7974	0.1712	4.6578	0.0000	0.4618	1.1329
fever_woe	0.5048	0.3153	1.6009	0.1094	-0.1132	1.1228
hypertension_woe	0.3149	0.2422	1.3002	0.1935	-0.1598	0.7896
respiratory problems_woe	0.5425	0.2983	1.8185	0.0690	-0.0422	1.1272
sex_woe	1.8598	2.3989	0.7752	0.4382	-2.8420	6.5616
pneumonia_woe	0.3237	0.2760	1.1729	0.2408	-0.2172	0.8647

Table 5.4: Results: Generalized linear model

Model:	GLM	AIC:	109.6265
Link Function:	logit	BIC:	-626.8618

	Coef.	Std.Err.	z	P> |z|	[0.025	0.975]
const	0.6516	0.2856	2.2819	0.0225	0.0919	1.2113
cough_woe	0.0907	0.3867	0.2346	0.8145	-0.6673	0.8487
age_woe	0.7839	0.1666	4.7060	0.0000	0.4574	1.1104
fever_woe	0.4863	0.3110	1.5638	0.1179	-0.1232	1.0959
hypertension_woe	0.3043	0.2395	1.2709	0.2038	-0.1650	0.7737
respiratory problems_woe	0.5505	0.2988	1.8422	0.0655	-0.0352	1.1362
pneumonia_woe	0.3547	0.2733	1.2976	0.1944	-0.1810	0.8904

Table 5.5: Results: Generalized linear model (sex dropped)

Model:	GLM	AIC:	107.4870
Link Function:	logit	BIC:	-635.0358

	Coef.	Std.Err.	z	P> |z|	[0.025	0.975]
const	0.6144	0.2736	2.2455	0.0247	0.0781	1.1507
pneumonia_woe	0.4567	0.2482	1.8400	0.0658	-0.0298	0.9433
respiratory problems_woe	0.5246	0.2942	1.7836	0.0745	-0.0519	1.1012
age_woe	0.8393	0.1634	5.1360	0.0000	0.5190	1.1596
fever_woe	0.5815	0.3037	1.9146	0.0555	-0.0138	1.1768

Table 5.6: Results: Generalized linear model (cough & hypertension dropped)

Seen below is the confusion matrices for each model. From these it can be seen that Model 2 (sex dropped) appears to be categorizing the test data the best with only 4.69% incorrect where as Model 3 (cough and hypertension dropped) is performing the worst with 14.06% categorized incorrectly. Considering that the AIC is similar in all cases and the type of data this is, the choice of model I would go with would be based on accuracy and as such would go with Model 2.

		True	
		0	1
Pred	0	32.81	3.12
	1	4.69	59.38

Model 1

		True	
		0	1
Pred	0	35.94	0
	1	4.69	59.38

Model 2

		True	
		0	1
Pred	0	28.12	7.81
	1	6.25	57.81

Model 3

Table 5.7: Confusion Matrices (Percentages)

Chapter 6

Conclusion

Appendices

Appendix A

Definintions

Origination. The period from when an client applies for a loan to its distribution of funds from the loan.

Piggyback Lending. An additional loan taken out on a property on top of the first mortgage.

Subprime Loan. A loan with higher interest rates, often given to clients with low credit score unable to qualify for normal rates.

Goods. The term to define a good client, most often meaning a client which does not default on a loan.

Bads. The term to define a bad client, most often meaning a client which ends up defaulting on a loan.

Bad Rate. The percentage of clients that have defaulted.

Appendix B

Tables

	BAD	LOAN	MORTDUE	VALUE	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO	DEBTINC
count	5621.0	5621.00	5278.00	5537.00	5294.00	5206.00	5356.00	5549.00	5422.00	5621.00	4447.00
mean	0.2	18846.02	73977.01	103025.40	9.00	0.24	0.45	179.77	1.19	21.45	34.07
std	0.4	11301.47	44813.54	58002.35	7.61	0.80	1.13	85.70	1.73	10.13	8.47
min	0.0	1100.00	2063.00	8000.00	0.00	0.00	0.00	0.00	0.00	0.00	0.52
25%	0.0	11300.00	46385.00	66922.00	3.00	0.00	0.00	115.57	0.00	15.00	29.43
50%	0.0	16500.00	65000.00	90008.00	7.00	0.00	0.00	173.63	1.00	20.00	35.02
75%	0.0	23500.00	91989.25	120724.00	13.00	0.00	0.00	230.72	2.00	26.00	39.14
max	1.0	89900.00	399550.00	855909.00	41.00	10.00	15.00	1168.23	17.00	71.00	203.31

Figure B.1: Summary Before Outliers Removed

	BAD	LOAN	MORTDUE	VALUE	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO
mean	0.18	17895.67	67997.48	97757.46	8.54	0.15	0.31	174.71	1.05	20.67
std	0.38	9543.89	36628.32	45795.23	6.89	0.48	0.76	75.72	1.34	9.00
min	0.00	1100.00	2063.00	8000.00	0.00	0.00	0.00	0.51	0.00	0.00
25%	0.00	11000.00	43416.50	65783.00	3.00	0.00	0.00	115.08	0.00	14.00
50%	0.00	16100.00	62344.00	88710.50	7.00	0.00	0.00	170.65	1.00	20.00
75%	0.00	22800.00	86782.50	117303.00	12.00	0.00	0.00	224.57	2.00	26.00
max	1.00	60500.00	207687.00	270794.00	29.00	3.00	4.00	397.87	7.00	48.00

Figure B.2: Summary After Cleaning

	LOAN	MORTDUE	VALUE	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO
BAD	-0.105	-0.0842	-0.1049	-0.0620	0.2175	0.2858	-0.1820	0.1369	-0.0632
LOAN		0.1730	0.3045	0.0453	0.0036	-0.0946	0.1172	0.0661	0.1117
MORTDUE			0.8994	-0.0693	-0.0432	-0.0424	0.1065	-0.0061	0.3389
VALUE				-0.0160	-0.0570	-0.0518	0.1775	-0.0267	0.3107
YOJ					-0.0464	0.0341	0.1669	-0.0488	0.0307
DEROG						0.1680	-0.0614	0.1249	0.0060
DELINQ							-0.0108	0.0298	0.1101
CLAGE								-0.0906	0.2202
NINQ									0.1046

Figure B.3: Correlation Table

Model:	GLM	AIC:	2543.2467
Link Function:	logit	BIC:	-25758.5600
Dependent Variable:	BAD	Log-Likelihood:	-1260.6

	Coef.	Std.Err.	z	P> |z|	[0.025	0.975]
const	-1.5661	0.0533	-29.4073	0.0000	-1.6705	-1.4617
CLNO_woe	0.8257	0.1332	6.1991	0.0000	0.5646	1.0867
YOJ_woe	0.8134	0.1973	4.1221	0.0000	0.4267	1.2002
REASON_woe	-0.4355	0.3941	-1.1049	0.2692	-1.2080	0.3370
LOAN_woe	0.8959	0.1003	8.9325	0.0000	0.6994	1.0925
JOB_woe	0.8907	0.1612	5.5243	0.0000	0.5747	1.2067
DELINQ_woe	1.0305	0.0786	13.1048	0.0000	0.8764	1.1846
DEROG_woe	0.7534	0.1047	7.1983	0.0000	0.5482	0.9585
MORTDUE_woe	0.6597	0.1889	3.4922	0.0005	0.2894	1.0299
NINQ_woe	1.0383	0.1520	6.8321	0.0000	0.7404	1.3362
CLAGE_woe	0.9666	0.1002	9.6503	0.0000	0.7703	1.1629

Table B.1: Results: Model with VALUE dropped

Model:	GLM	AIC:	2495.4175
Link Function:	logit	BIC:	-25800.2346
Dependent Variable:	BAD	Log-Likelihood:	-1235.7

	Coef.	Std.Err.	z	P> |z|	[0.025	0.975]
const	-1.5802	0.0542	-29.1318	0.0000	-1.6865	-1.4738
DELINQ_woe	1.0393	0.0793	13.1121	0.0000	0.8839	1.1946
CLAGE_woe	0.9408	0.1010	9.3183	0.0000	0.7429	1.1387
NINQ_woe	0.9875	0.1533	6.4408	0.0000	0.6870	1.2880
MORTDUE_woe	-0.1417	0.2347	-0.6037	0.5461	-0.6016	0.3183
CLNO_woe	0.8104	0.1347	6.0148	0.0000	0.5463	1.0744
LOAN_woe	0.8631	0.1002	8.6108	0.0000	0.6667	1.0596
REASON_woe	-0.6331	0.4029	-1.5714	0.1161	-1.4229	0.1566
JOB_woe	0.8575	0.1634	5.2485	0.0000	0.5373	1.1778
VALUE_woe	0.8590	0.1537	5.5879	0.0000	0.5577	1.1603
DEROG_woe	0.7875	0.1050	7.4994	0.0000	0.5817	0.9933
YOJ_woe	0.8890	0.1665	5.3401	0.0000	0.5627	1.2152

Table B.2: Results: Model 2

Appendix C

Python Code

```
# -*- coding: utf-8 -*-
"""
Created on Fri Aug  7 19:16:46 2020

@author: Alexander
"""
import numpy as np
import pandas as pd
import statsmodels.api as sm

# Read data
df = pd.read_csv("E:/GitHub/Credit-Scorecard-Project/Python/hmeq.csv")

# Remove the top 1% quantile from each variable
cols = df.select_dtypes([np.number]).columns
cols = cols[1:]
for col in cols:
    df = df.loc[
        (df[col] < df[col].quantile(0.99))
        | (df[col].isna())
    ]

# Get summary details
t = df.describe()

## Mortgage impute
df = sm.add_constant(df, has_constant='add')

mort_missing_with_value = df.loc[
    (df.MORTDUE.isna())
    & ~(df.VALUE.isna())
]
mort_with_value = df.loc[
    ~(df.MORTDUE.isna())
    & ~(df.VALUE.isna())
]

mort_ols = sm.OLS(mort_with_value.MORTDUE,
                  mort_with_value[['const', 'VALUE']])
mort_fit = mort_ols.fit()

predictions = mort_fit.predict(
    mort_missing_with_value[['const', 'VALUE']]
)

# Get mean before updating
mort_mean = df.MORTDUE.mean()

df.MORTDUE.update(predictions)
# Fill any remaining with original mean
df.MORTDUE.fillna(mort_mean, inplace=True)

## Value impute
value_missing_with_mort = df.loc[
    (df.VALUE.isna())
    & ~(df.MORTDUE.isna())
]
value_with_mort = df.loc[
    ~(df.VALUE.isna())
    & ~(df.MORTDUE.isna())
]

value_ols = sm.OLS(value_with_mort.VALUE,
                   value_with_mort[['const', 'MORTDUE']])
value_fit = value_ols.fit()

predictions = value_fit.predict(
    value_missing_with_mort[['const', 'MORTDUE']]
)

# Get mean before updating
value_mean = df.VALUE.mean()

df.VALUE.update(predictions)
# Fill any remaining with original mean
df.VALUE.fillna(value_mean, inplace=True)

# Drop constant column
df.drop('const', axis=1, inplace=True)

df.iloc[:, :-1] = df.iloc[:, :-1].fillna(df.median())

# Drop DEBTINC
```

```
df = df.drop('DEBTINC', axis=1)

# Drop any remaining missing values
df.dropna(inplace=True)

# 0 missing values remaining
df.isna().sum()

# Save cleaned data
df.to_csv(
    r"E:\GitHub\Credit-Scorecard-Project\Python\hmeq_clean.csv",
    index=False
)

# -*- coding: utf-8 -*-
"""
Created on Tue Aug 11 18:55:45 2020

@author: Alexander
"""
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from scipy import stats

# Read data
df = pd.read_csv("E:/GitHub/Credit-Scorecard-Project/Python/hmeq_clean.csv")

# Get summary data
t = df.describe()

# Get correlation table
corr_table = df.corr()
corr_table = corr_table.mask(
    np.tril(np.ones(corr_table.shape)).astype(np.bool)
)

sns.set(style="whitegrid")

## Loan Plot
fig, (ax_dist, ax_box_good, ax_box_bad) = (
    plt.subplots(3,
                 sharex=True,
                 gridspec_kw={"height_ratios": (.80, .10, .10)},
                 figsize=[15,8])
)
```

```
# Add a graph in each part
goods = df.loc[df.BAD == 0, 'LOAN']
bads = df.loc[df.BAD == 1, 'LOAN']

sns.distplot(
    goods,
    hist=False,
    label='Good',
    color='green',
    ax=ax_dist
)

sns.distplot(
    bads,
    hist=False,
    label='Bad',
    color='red',
    ax=ax_dist
).tick_params(labelsize=16)

sns.boxplot(goods, ax=ax_box_good, color='green')
sns.boxplot(bads, ax=ax_box_bad, color='red').tick_params(labelsize=16)

# Remove x axis name for the boxplot
ax_box_good.set(xlabel='')
ax_box_bad.set(xlabel='')

plt.savefig('figs/loan_dist.pdf')

## Mortdue Plot
fig, (ax_dist, ax_box_good, ax_box_bad) = (
    plt.subplots(3,
                 sharex=True,
                 gridspec_kw={"height_ratios": (.80, .10, .10)},
                 figsize=[15,8])
)

# Add a graph in each part
goods = df.loc[df.BAD == 0, 'MORTDUE']
bads = df.loc[df.BAD == 1, 'MORTDUE']
sns.distplot(
    goods,
    hist=False,
    label='Good',
    color='green',
    ax=ax_dist
)
sns.distplot(
    bads,
    hist=False,
    label='Bad',
    color='red',
    ax=ax_dist
).tick_params(labelsize=16)

sns.boxplot(goods, ax=ax_box_good, color='green')
sns.boxplot(bads, ax=ax_box_bad, color='red').tick_params(labelsize=16)

# Remove x axis name for the boxplot
ax_box_good.set(xlabel='')
ax_box_bad.set(xlabel='')

plt.savefig('figs/mortdue_dist.pdf')

## Value Plot
fig, (ax_dist, ax_box_good, ax_box_bad) = (
    plt.subplots(3,
                 sharex=True,
                 gridspec_kw={"height_ratios": (.80, .10, .10)},
                 figsize=[15,8])
)
# Add a graph in each part
goods = df.loc[df.BAD == 0, 'VALUE']
bads = df.loc[df.BAD == 1, 'VALUE']
sns.distplot(
    goods,
    hist=False,
    label='Good',
    color='green',
    ax=ax_dist
)
sns.distplot(
    bads,
    hist=False,
    label='Bad',
    color='red',
    ax=ax_dist
).tick_params(labelsize=16)

sns.boxplot(goods, ax=ax_box_good, color='green')
sns.boxplot(bads, ax=ax_box_bad, color='red').tick_params(labelsize=16)

# Remove x axis name for the boxplot
ax_box_good.set(xlabel='')
ax_box_bad.set(xlabel='')

plt.savefig('figs/value_dist.pdf')
```

```
# REASON plot
reason_perc = df.groupby('REASON')['BAD'].value_counts(normalize=True)
reason_perc = reason_perc * 100
reason_perc = reason_perc.rename('Perc').reset_index()

fig, ax = plt.subplots(figsize=[13.5,5])
sns.barplot(
    x='REASON',
    y='Perc',
    hue='BAD',
    data=reason_perc,
    palette=['green', 'red'],
    ax=ax
)

ax.set(ylim=(0,100))
for p in ax.patches:
    txt = str(p.get_height().round(2)) + '%'
    txt_x = p.get_x() + 0.1
    txt_y = p.get_height() + 1
    ax.text(txt_x,txt_y,txt)

plt.savefig('figs/reason_cat.pdf')

# JOB plot
reason_perc = df.groupby('JOB')['BAD'].value_counts(normalize=True)
reason_perc = reason_perc * 100
reason_perc = reason_perc.rename('Perc').reset_index()

fig, ax = plt.subplots(figsize=[15,5])
sns.barplot(
    x='JOB',
    y='Perc',
    hue='BAD',
    data=reason_perc,
    palette=['green', 'red'],
    ax=ax
)
ax.set(ylim=(0,100))

for p in ax.patches:
    txt = str(p.get_height().round(2)) + '%'
    txt_x = p.get_x() + 0.05
    txt_y = p.get_height() + 1
    ax.text(txt_x,txt_y,txt)

plt.savefig('figs/job_cat.pdf')

## YOJ Plot
```

```
fig, (ax_dist, ax_box_good, ax_box_bad) = (
    plt.subplots(3,
                 sharex=True,
                 gridspec_kw={"height_ratios": (.80, .10, .10)},
                 figsize=[15,8]
                 )
)
# Add a graph in each part
goods = df.loc[df.BAD == 0, 'YOJ']
bads = df.loc[df.BAD == 1, 'YOJ']
sns.distplot(
    goods,
    hist=False,
    label='Good',
    color='green',
    ax=ax_dist
)
sns.distplot(
    bads,
    hist=False,
    label='Bad',
    color='red',
    ax=ax_dist
).tick_params(labelsize=16)

sns.boxplot(goods, ax=ax_box_good, color='green')
sns.boxplot(bads, ax=ax_box_bad, color='red').tick_params(labelsize=16)

# Remove x axis name for the boxplot
ax_box_good.set(xlabel='')
ax_box_bad.set(xlabel='')

plt.savefig('figs/yoj_dist.pdf')

# YOJ plot
fig, ax = plt.subplots(figsize=[13.5,8])
sns.boxplot(x='BAD', y='YOJ', data=df, palette=['green', 'red'], ax=ax)
plt.savefig('figs/yoj_box.pdf')

# DEROG plot
derog_perc = df.groupby('DEROG')['BAD'].value_counts(normalize=True)
derog_perc = derog_perc * 100
derog_perc = derog_perc.rename('Perc').reset_index()

fig, ax = plt.subplots(figsize=[13.5,5])
sns.barplot(
    x='DEROG',
    y='Perc',
    hue='BAD',
    data=derog_perc,
    palette=['green', 'red'],
    ax=ax
)

for p in ax.patches:
    txt = str(p.get_height().round(2)) + '%'
    txt_x = p.get_x() + 0.08
    txt_y = p.get_height() + 1
    ax.text(txt_x,txt_y,txt)

plt.savefig('figs/derog_cat.pdf')

# DELINQ plot
derog_perc = df.groupby('DELINQ')['BAD'].value_counts(normalize=True)
derog_perc = derog_perc * 100
derog_perc = derog_perc.rename('Perc').reset_index()

fig, ax = plt.subplots(figsize=[13.5,5])
sns.barplot(
    x='DELINQ',
    y='Perc',
    hue='BAD',
    data=derog_perc,
    palette=['green', 'red'],
    ax=ax
)

for p in ax.patches:
    txt = str(p.get_height().round(2)) + '%'
    txt_x = p.get_x() + 0.08
    txt_y = p.get_height() + 1
    ax.text(txt_x,txt_y,txt)

plt.savefig('figs/delinq_cat.pdf')

## CLAGE Plot
fig, (ax_dist, ax_box_good, ax_box_bad) = (
    plt.subplots(3,
                 sharex=True,
                 gridspec_kw={"height_ratios": (.80, .10, .10)},
                 figsize=[15,8])
)
# Add a graph in each part
goods = df.loc[df.BAD == 0, 'CLAGE']
bads = df.loc[df.BAD == 1, 'CLAGE']
sns.distplot(
    goods,
    hist=False,
    label='Good',
    color='green',
    ax=ax_dist
)
sns.distplot(
    bads,
    hist=False,
    label='Bad',
    color='red',
    ax=ax_dist
).tick_params(labelsize=16)

sns.boxplot(goods, ax=ax_box_good, color='green')
sns.boxplot(bads, ax=ax_box_bad, color='red').tick_params(labelsize=16)

# Remove x axis name for the boxplot
ax_box_good.set(xlabel='')
ax_box_bad.set(xlabel='')

plt.savefig('figs/clage_dist.pdf')

# NINQ plot
ninq_perc = df.groupby('NINQ')['BAD'].value_counts(normalize=True)
ninq_perc = ninq_perc * 100
ninq_perc = ninq_perc.rename('Perc').reset_index()

fig, ax = plt.subplots(figsize=[13.5,5])
sns.barplot(
    x='NINQ',
    y='Perc',
    hue='BAD',
    data=ninq_perc,
    palette=['green', 'red'],
    ax=ax
)

for p in ax.patches:
    txt = str(p.get_height().round(2)) + '%'
    txt_x = p.get_x() + 0.05
    txt_y = p.get_height() + 1
    ax.text(txt_x,txt_y,txt)

plt.savefig('figs/ninq_cat.pdf')

## CLNO Plot
fig, (ax_dist, ax_box_good, ax_box_bad) = (
    plt.subplots(3,
                 sharex=True,
                 gridspec_kw={"height_ratios": (.80, .10, .10)},
                    figsize=[15,8])
)
# Add a graph in each part
goods = df.loc[df.BAD == 0, 'CLNO']
bads = df.loc[df.BAD == 1, 'CLNO']
sns.distplot(
    goods,
    hist=False,
    label='Good',
    color='green',
    ax=ax_dist
)
sns.distplot(
    bads,
    hist=False,
    label='Bad',
    color='red',
    ax=ax_dist
).tick_params(labelsize=16)
sns.boxplot(goods, ax=ax_box_good, color='green')
sns.boxplot(bads, ax=ax_box_bad, color='red').tick_params(labelsize=16)

# Remove x axis name for the boxplot
ax_box_good.set(xlabel='')
ax_box_bad.set(xlabel='')

plt.savefig('figs/clno_dist.pdf')


# JOB vs YOJ Plot
df['Count'] = df['YOJ'].groupby(df['JOB']).transform('mean')
df = df.sort_values('Count')

fig, ax = plt.subplots(figsize=[13.5,5])
plt_ = sns.barplot(x='JOB', y='YOJ', data=df, ax=ax)
plt_.tick_params(labelsize=16)
plt_.set_xlabel('JOB', fontsize=20)
plt_.set_ylabel('YOJ', fontsize=20)
plt.savefig('figs/job_yoj_cat.pdf')

# -*- coding: utf-8 -*-
"""
Created on Wed Aug 12 20:01:18 2020

@author: Alexander
"""
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
```

```python
import scorecardpy as sc
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

# Read data
df = pd.read_csv("E:\GitHub\Credit-Scorecard-Project\Python\hmeq_clean.csv")

# df.drop(['VALUE'], axis=1, inplace=True)

# Apply Transformations
df.LOAN = np.log(df.LOAN)
# df.MORTDUE = np.log(df.MORTDUE)
# df.VALUE = np.log(df.VALUE)
# Variable contained zeros so added 1 year to every observation
df.YOJ = np.log(df.YOJ + 1)

# Drop REASON and MORTDUE
df.drop(['REASON', 'MORTDUE'], axis=1, inplace=True)

# Create WOE bins
bins = sc.woebin(df, 'BAD', method='chimerge')

# Job was not binning correctly, this fixed that
break_list = {'JOB': df.JOB.unique().tolist()}
job_bins = sc.woebin(df, 'BAD', method='chimerge', x=['JOB'],
                     breaks_list=break_list)
bins['JOB'] = job_bins['JOB']

# Plot WOE bins
# fig, axs = plt.subplots(ncols=2)
# sc.woebin_plot(bins, figsize=[8,5])

# Print results of binning
# for k, bin_ in bins.items():
#     print(bins[k].iloc[:,0:-2].round(2).to_latex(index=False))

# split into train and test set
train, test = sc.split_df(df, 'BAD').values()

# Convert values into woe
train_woe = sc.woebin_ply(train, bins)
test_woe = sc.woebin_ply(test, bins)

# Add constant
train_woe = sm.add_constant(train_woe)
test_woe = sm.add_constant(test_woe)

y_train = train_woe.loc[:, 'BAD']
```

```
X_train = train_woe.loc[:,train_woe.columns != 'BAD']
y_test = test_woe.loc[:,'BAD']
X_test = test_woe.loc[:,train_woe.columns != 'BAD']

# Fit logit model
lr = sm.GLM(y_train, X_train, family=sm.families.Binomial())
fit = lr.fit()

fit.summary()

# Get probabilities
train_pred = fit.predict(X_train)
test_pred = fit.predict(X_test)

# Plot diagnositcs
test_perf = sc.perf_eva(y_test, test_pred, title = "test", plot_type=['ks'])
test_perf = sc.perf_eva(y_test, test_pred, title = "test", plot_type=['roc'])

class ModelDetails():

    def __init__(self, intercept, coefs):
        """Because the scorecardpy package can only take a model class of
        LogisticRegression from the scikit-learn package this class is needed
        to hold the values from the statsmodels package.
        """

        self.intercept_ = [intercept]
        self.coef_ = [coefs.tolist()]

model = ModelDetails(fit.params[0], fit.params[1:])

# Create scorecard
card = sc.scorecard(bins, model, X_train.columns[1:], points0=800, pdo=50)

# Create scores
train_score = sc.scorecard_ply(train, card, print_step=0)
test_score = sc.scorecard_ply(test, card, print_step=0)

# Plot scorecard
sc.perf_psi(
  score = {'train':train_score, 'test':test_score},
  label = {'train':y_train, 'test':y_test}
)


df['SCORE'] = 0
df.SCORE.update(train_score.score)

## Score Train Plot
```

```
fig, (ax_dist, ax_box_good, ax_box_bad) = (
    plt.subplots(3,
                 sharex=True,
                 gridspec_kw={"height_ratios": (.80, .10, .10)},
                 figsize=[15,8])
)
# Add a graph in each part
subset = df.loc[df.SCORE != 0]
goods = subset.loc[subset.BAD == 0, 'SCORE']
bads = subset.loc[subset.BAD == 1, 'SCORE']
sns.distplot(
    goods, hist=False, label='Good', color='green', ax=ax_dist)
sns.distplot(
    bads, hist=False, label='Bad', color='red', ax=ax_dist
).tick_params(labelsize=16)
sns.boxplot(goods, ax=ax_box_good, color='green')
sns.boxplot(bads, ax=ax_box_bad, color='red').tick_params(labelsize=16)
ax_dist.set_xlabel("SCORE: TRAIN",fontsize=20)

# Remove x axis name for the boxplot
ax_box_good.set(xlabel='')
ax_box_bad.set(xlabel='')

divergence_train = (
    pow((goods.mean() - bads.mean()), 2)
    / (0.5 * (goods.var() + bads.var()))
)

df['SCORE'] = 0
df.SCORE.update(test_score.score)

## Score Test Plot
fig, (ax_dist, ax_box_good, ax_box_bad) = (
    plt.subplots(3,
                 sharex=True,
                 gridspec_kw={"height_ratios": (.80, .10, .10)},
                 figsize=[15,8])
)
# Add a graph in each part
subset = df.loc[df.SCORE != 0]
goods = subset.loc[subset.BAD == 0, 'SCORE']
bads = subset.loc[subset.BAD == 1, 'SCORE']
sns.distplot(
    goods, hist=False, label='Good', color='green', ax=ax_dist)
sns.distplot(
    bads, hist=False, label='Bad', color='red', ax=ax_dist
).tick_params(labelsize=16)
sns.boxplot(goods, ax=ax_box_good, color='green')
sns.boxplot(bads, ax=ax_box_bad, color='red').tick_params(labelsize=16)
```

```
ax_dist.set_xlabel("SCORE: TEST",fontsize=20)

# Remove x axis name for the boxplot
ax_box_good.set(xlabel='')
ax_box_bad.set(xlabel='')

divergence_test = (
    pow((goods.mean() - bads.mean()), 2)
    / (0.5 * (goods.var() + bads.var()))
)

# -*- coding: utf-8 -*-
"""
Created on Sun Aug 23 12:09:36 2020

@author: Alexander
"""
import numpy as np
import pandas as pd
import scorecardpy as sc
import statsmodels.api as sm

# # Subsetting
# dat = pd.read_csv("E:\GitHub\Credit-Scorecard-Project\Python\latestdata.csv")
# df = dat.loc[:, ['ID', 'age', 'sex', 'symptoms', 'chronic_disease', 'outcome']]
# df = df.dropna(subset=['outcome', 'age', 'sex', 'symptoms'])
# df.to_csv(
# 'E:\GitHub\Credit-Scorecard-Project\Python\covid_subset.csv', index=False
# )

df = pd.read_csv("E:\GitHub\Credit-Scorecard-Project\Python\covid_subset.csv")

df['symptoms'] = df['symptoms'].str.replace(':', ',')
df['symptoms'] = df['symptoms'].str.replace(';', ',')
df['symptoms'] = df['symptoms'].str.replace(', ', ',')
df['symptoms'] = df['symptoms'].str.lower()

df['symptoms'] = df['symptoms'].str.replace(
    'acute respiratory disease syndrome', 'respiratory problems')
df['symptoms'] = df['symptoms'].str.replace(
    'acute respiratory distress syndrome', 'respiratory problems')
df['symptoms'] = df['symptoms'].str.replace(
    'acute respiratory disease', 'respiratory problems')
df['symptoms'] = df['symptoms'].str.replace(
    'acute respiratory distress', 'respiratory problems')
df['symptoms'] = df['symptoms'].str.replace(
    'acute respiratory failure', 'respiratory problems')
df['symptoms'] = df['symptoms'].str.replace(
    'respiratory stress', 'respiratory problems')
```

```
df['symptoms'] = df['symptoms'].str.replace(
    'respiratory symptoms', 'respiratory problems')
df['symptoms'] = df['symptoms'].str.replace(
    'severe acute respiratory infection', 'respiratory problems')

df['symptoms'] = df['symptoms'].str.replace(
    'severe pneumonia', 'pneumonia')
df['symptoms'] = df['symptoms'].str.replace(
    'dry cough', 'cough')

sym = df['symptoms'].str.get_dummies(sep=',')
t = sym.sum()

df['chronic_disease'] = df['chronic_disease'].str.replace(':', ',')
df['chronic_disease'] = df['chronic_disease'].str.replace(';', ',')
df['chronic_disease'] = df['chronic_disease'].str.replace(', ', ',')
df['chronic_disease'] = df['chronic_disease'].str.lower()

df['chronic_disease'] = df['chronic_disease'].str.replace(
    ' for more than 20 years', ',')
df['chronic_disease'] = df['chronic_disease'].str.replace(
    ' surgery four years ago', ',')
df['chronic_disease'] = df['chronic_disease'].str.replace(
    ' accident ', ',')
df['chronic_disease'] = df['chronic_disease'].str.replace(
    ' for five years', ',')
df['chronic_disease'] = df['chronic_disease'].str.replace(
    'hypertensive', 'hypertension')
df['chronic_disease'] = df['chronic_disease'].str.replace(
    'hypertenstion', 'hypertension')

cd = df['chronic_disease'].str.get_dummies(sep=',')
v = cd.sum()

sym = sym.loc[:, sym.sum() >= 40]
cd = cd.loc[:, cd.sum() >= 40]

df = pd.concat([df, sym], axis=1)
df = pd.concat([df, cd], axis=1)

df['outcome'] = df['outcome'].str.lower()
df.outcome = np.where(
    df.outcome.isin(['death', 'dead', 'died', 'deceased']),
    1,
    0
)

df.sex = np.where(df.sex == 'male',
         1,
        0
)

df.drop(['ID', 'symptoms', 'chronic_disease'], axis=1, inplace=True)
df.age = df.age.str.replace('50-59', '55')
df.age = df.age.str.replace('60-69', '65')
df.age = df.age.str.replace('15-88', '45')
df.age = df.age.str.replace('80-89', '85')
df.age = df.age.str.replace('20-29', '25')
df.age = df.age.str.replace('40-49', '45')
df.age = df.age.str.replace('70-79', '75')
df.age = df.age.str.replace('90-99', '95')
df.age = df.age.str.replace('28-35', '31')
df.age = df.age.str.replace('80-', '80')

df.age = df.age.astype('float')
df.dropna(inplace=True)

# Reorder columns
cols = df.columns.tolist()
cols = cols[2:3] + cols[0:2] + cols[3:]

df = df[cols]

df.drop(['diabetes' ], axis=1, inplace=True)

bins = sc.woebin(df, 'outcome', method='chimerge')

cols = df.iloc[:, 2:].columns
break_list = {}
for col in cols:
    break_list[col] = [1.0]

bins.update(sc.woebin(df, 'outcome', method='chimerge', x=cols.tolist(),
                      breaks_list=break_list))

# split into train and test set
train, test = sc.split_df(df, 'outcome').values()

# Convert values into woe
train_woe = sc.woebin_ply(train, bins)
test_woe = sc.woebin_ply(test, bins)

train_woe = sm.add_constant(train_woe)
test_woe = sm.add_constant(test_woe)

y_train = train_woe.loc[:,'outcome']
X_train = train_woe.loc[:,train_woe.columns != 'outcome']
y_test = test_woe.loc[:,'outcome']
```

```
X_test = test_woe.loc[:,train_woe.columns != 'outcome']

# Fit logit model
lr = sm.GLM(y_train, X_train, family=sm.families.Binomial())
fit = lr.fit()

fit.summary()

pred = np.array(fit.predict(X_test) > 0.5, dtype=float)
table = np.histogram2d(y_test, pred, bins=2)[0]
table = np.round(( table / table.sum() ) * 100, 2)
```

Bibliography

Baesens, Bart, Daniel Roesch, and Harald Scheule (2016). *Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS*. John Wiley & Sons.

Bailey, Murray (2004). *Credit scoring: The principles and practicalities*. White Box Publ.

Buhi, Eric R, Patricia Goodson, and Torsten B Neilands (2008). "Out of sight, not out of mind: Strategies for handling missing data". In: *American journal of health behavior* 32.1, pp. 83–92.

Canner, Glenn B, Thomas A Durkin, and Charles A Luckett (1998). "Recent developments in home equity lending". In: *Fed. Res. Bull.* 84, p. 241.

CoreLogic (2016). "Home Equity Lending Landscape". In:

Group, Open COVID-19 Data Working (2020). *Detailed Epidemiological Data from the COVID-19 Outbreak*. Accessed on yyyy-mm-dd from `http://virological.org/t/epidemiological-data-from-the-ncov-2019-outbreak-early-descriptions-from-publicly-available-data/337`.

Kerber, Randy (1992). "Chimerge: Discretization of numeric attributes". In: *Proceedings of the tenth national conference on Artificial intelligence*, pp. 123–128.

Newman, Daniel A (2014). "Missing data: Five practical guidelines". In: *Organizational Research Methods* 17.4, pp. 372–411.

Nyitrai, Tamás and Miklós Virág (2019). "The effects of handling outliers on the performance of bankruptcy prediction models". In: *Socio-Economic Planning Sciences* 67, pp. 34–42.

Python Package for Statistical Modelling. URL: `https://github.com/statsmodels/statsmodels`.

ShichenXie. *Python Package for Credit Scoring*. URL: `https://github.com/ShichenXie/scorecardpy`.

Tape, TG (2000). "Using the Receiver Operating Characteristic (ROC) curve to analyze a classification model". In: *University of Nebraska*, pp. 1–3.

Weicher, John C (1997). *The Home Equity Lending Industry: Refinancing Mortgages for Borrowers with Impaired Credit*. Hudson Inst.

Yurdakul, Bilal (2018). "Statistical properties of population stability index". In: