Credit Card Default Model

Model Development Guide

PREDICT 498 Capstone with Dr. Bhatti, Winter 2018

M.S. Predictive Analytics

Northwestern University

Author: Ahmed Mustakim

Date: February 26, 2018

Contents

1. Introduction

In this capston project paper, we will analyze a data set that contains the credit card payment history and socio demographic attributes of a sample group of customers in a financial institution. We will then predict which of these customers will default in their credit card payments using four different predictive modeling methods that vary in complexity but are all implemented in the R statistical language. Prior to constructing those models, we will apply a set of universal data cleansing rules on the data based on a close reading of the data dictionary and then engineer new features that will allow us to normalize and benchmark payment, credit utilization and other metrics across individual customer records. Baseline predictive models such as these can inform the basic policy decisions in a consumer credit organization and it also enables students in this degree program to understand how machine learning techniques are applied in the real world.

2. The Data

The data for this capstone project contains the credit card payment history of 30,000 Taiwanese customers of a financial institution for 6 consecutive months from April 2005. This data set is now maintained in University of California Irvine's Machine Learning Repository. It was originally assembled to measure the predictive accuracy of different statistical models which tried to predict if customers would default on their monthly credit card payments based on payment history and socio-demographic attributes.

There are 24 attributes in this data set, excluding an "ID" attribute which is used to identify a unique observation (out of 30,000 observations) about an individual customer. Out of these 24 attributes, the "DEFAULT" attribute indicates whether the customer actually defaulted

on their next month's payment – this is the dependent or response (Y) variable. The remaining 23 attributes are independent or explanatory (X) variables and they contain payment and billing history for the past 6 months (from April 2005 to September 2005) along with certain socio-demographic information about each of the 30,000 customers like age, education level, marital status and the credit limit allocated for that customer.

The data dictionary below provides a full accounting of all the attributes that are available in the data set along with some pertinent summary statistics about each attribute or field:

Table 2.1: *Data Dictionary (note: 1 New Taiwanese Dollar is approximately 3 U.S. cents)*

Attribute Name	Attribute Description	Data type	Min	Max	Mean	Median
ID	Unique record (or observation) identifier (not considered one of the official 24 attributes)	integer	1	30,000	15,000	15,000
LIMIT_BAL	Amount of the given credit (New Taiwanese dollar). It includes both the individual consumer credit and his/her family (supplementary) credit.	integer (discrete)	10,000	1,000,000	167,484	140,000
SEX	Gender (1 = male; 2 = female)	integer (binary)	1	2	1.6	2
EDUCATION	Education (1 = graduate school; 2 = university; 3 = high school; 4 = others)	integer (categorical)	0	6	1.9	2
MARRIAGE	Marital status (1 = married; 2 = single; 3 = others)	integer (categorical)	0	3	1.6	2
AGE	Age (year)	integer (discrete)	21	79	35.5	34
PAY_0	Repayment status in September 2005: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; … 8 = payment delay for eight months; 9 = payment delay for nine months and above.	integer (categorical)	-2	8	-0.017	0
PAY_2	Repayment status in August 2005: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . . 8 = payment delay for eight months; 9 = payment delay for nine months and	integer (categorical)	-2	8	-0.134	0

Attribute Name	Attribute Description	Data type	Min	Max	Mean	Median
	above.					
PAY_3	Repayment status in July 2005: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; … 8 = payment delay for eight months; 9 = payment delay for nine months and above.	integer (categorical)	-2	8	-0.166	0
PAY_4	Repayment status in June 2005: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; … 8 = payment delay for eight months; 9 = payment delay for nine months and above.	integer (categorical)	-2	8	-0.221	0
PAY_5	Repayment status in May 2005: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; … 8 = payment delay for eight months; 9 = payment delay for nine months and above.	integer (categorical)	-2	8	-0.266	0
PAY_6	Repayment status in April 2005: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; … 8 = payment delay for eight months; 9 = payment delay for nine months and above.	integer (categorical)	-2	8	-0.291	0
BILL_AMT1	Amount of bill statement (New Taiwanese dollar) in September 2005	integer (discrete)	-165,580	964,511	51,223	22,382
BILL_AMT2	Amount of bill statement (New Taiwanese dollar) in August 2005	integer (discrete)	-69,777	983,931	49,179	21,200
BILL_AMT3	Amount of bill statement (New Taiwanese dollar) in July 2005	integer (discrete)	-157,264	1,664,089	47,013	20,089
BILL_AMT4	Amount of bill statement (New Taiwanese dollar) in June 2005	integer (discrete)	-170,000	891,586	43,263	19,052
BILL_AMT5	Amount of bill statement (New Taiwanese dollar) in May 2005	integer (discrete)	-81,334	927,171	40,311	18,105

Attribute Name	Attribute Description	Data type	Min	Max	Mean	Median
BILL_AMT6	Amount of bill statement (New Taiwanese dollar) in April 2005	integer (discrete)	-339,603	961,664	38,872	17,071
PAY_AMT1	Amount paid (New Taiwanese dollar) in September 2005	integer (discrete)	0	873,552	5,664	2,100
PAY_AMT2	Amount paid (New Taiwanese dollar) in August 2005	integer (discrete)	0	1,684,259	5,921	2,009
PAY_AMT3	Amount paid (New Taiwanese dollar) in July 2005	integer (discrete)	0	896,040	5,226	1,800
PAY_AMT4	Amount paid (New Taiwanese dollar) in June 2005	integer (discrete)	0	621,000	4,826	1,500
PAY_AMT5	Amount paid (New Taiwanese dollar) in May 2005	integer (discrete)	0	426,529	4,799.40	1,500
PAY_AMT6	Amount paid (New Taiwanese dollar) in April 2005	integer (discrete)	0	528,666	5,215.50	1,500
DEFAULT	Did the customer default on the payment in October 2005 (1 = Yes, 0 = No)	integer (binary)	0	1	0.2	0

A closer look at the table above reveals some discrepancies between the "Attribute Description" column and the summary statistics (min, max) columns, which means that there are some potential data quality issues in this data set. For instance:

- The minimum EDUCATION level in this data set is 0 and the maximum is 6. However we expect EDUCATION levels to be between 1 and 4 where 1 = graduate school; 2 = university; 3 = high school; 4 = others.
- The minimum MARRIAGE status in the data is 0 but we expect MARRIAGE status to start at 1, not 0 where 1 = married; 2 = single; 3 = others.
- Similarly the minimum value for all the repayment status attributes (PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, and PAY_6) is -2 but we expect the minimum value to be -1 which means duly paid. In fact any value that is not -1, 1, 2, 3, 4, 5, 6, 7, 8 or 9

(where 1 means a payment delay of 1 month, 2 is a delay of 2 months and so on) are subjected to a systematic data quality process.

Unexpected values in range bound variables are therefore replaced with an 'NA' (meaning Not Available). For example EDUCATION values below 1 and above 4 are replaced with 'NA':

Figure 2.1: *Illustration of the data quality process using EDUCATION as an example*

This consistent data quality approach is applied to all the unexpected values in range-bound variable. However this rules is implemented judiciously and with nuance in mind. For instance all the six BILL_AMT variables (which reflect the monthly balance) contain negative values – this may be due to overpayment on the customer's part. In this scenario, the original values are retained.

To conduct the model validation and monitor performance of the model later, the data set is split into 3 sub-sets: a training set, a test set and a validation set. This is achieved by creating 3 binary (0 or 1) attributes (called "train", "test" and "validate") with the help of the uniform (0,1) normally distributed random variable "u". This allows us to subset into the desired set as needed. A complementary "data.group" attribute is also created and it contains the same information – its

values are 1, 2 or 3, where 1 means training set, 2 means test set and 3 means validation set. The following table shows how the original 30,000 observations are split among these 3 subsets:

Table 2.1: *Data sub-set counts*

Training	15,180 observations	50% of dataset
Test	7,323 observations	25% of dataset
Validation	7,497 observations	25% of dataset
TOTAL	**30,000 observations**	**100% of dataset**

3. Feature Engineering

In addition to the attributes described thus far, some new features are engineered using the existing attributes in the form of normalized rates which in turn sometimes leads to a better model, than using absolute values. These new features are described below (note that the scaling factor is intentionally kept consistent across all engineered features (i.e. 0 to 1), except for the "UTIL_VEL" and "TOTAL_UTIL_VEL" features which can be negative and therefore their range is from -1 to 1; also a division by a 0 denominator usually results in an error but here it is handled based on the value of the numerator):

Table 3.1: *Engineered Features*

Feature name	Description and generic formula	Exact formula	How it's scaled
UTIL_RT1	The credit utilization rate measures the percentage of available credit a consumer has used at a given point in time (in this case at the end of **September 2005**). It's calculated as follows: *Monthly balance / Credit Line.* Typically a higher utilization tends to have a negative impact on an overall 'FICO' type credit score and it may indicate a higher	*if* LIMIT_BAL = 0 (*if* BILL_AMT1 > 0 *then* 1 *else 0*) *else* (*if BILL_AMT1> LIMIT_BAL* *then 1*	0 to 1

Feature name	Description and generic formula	Exact formula	How it's scaled
	probability of default.	*else* BILL_AMT1 / LIMIT_BAL)	
UTIL_RT2	The credit utilization rate measures the percentage of available credit a consumer has used at a given point in time (in this case at the end of **August 2005**). It's calculated as follows: *Monthly balance / Credit Line.* Typically a higher utilization tends to have a negative impact on an overall 'FICO' type credit score and it may indicate a higher probability of default.	*if* LIMIT_BAL = 0 (*if* BILL_AMT2 > 0 *then* 1 *else 0*) *else* (*if BILL_AMT2> LIMIT_BAL* *then 1* *else* BILL_AMT2 / LIMIT_BAL)	0 to 1
UTIL_RT3	The credit utilization rate measures the percentage of available credit a consumer has used at a given point in time (in this case at the end of **July 2005**). It's calculated as follows: *Monthly balance / Credit Line.* Typically a higher utilization tends to have a negative impact on an overall 'FICO' type credit score and it may indicate a higher probability of default.	*if* LIMIT_BAL = 0 (*if* BILL_AMT3 > 0 *then* 1 *else 0*) *else* (*if BILL_AMT3> LIMIT_BAL* *then 1* *else* BILL_AMT3 / LIMIT_BAL)	0 to 1
UTIL_RT4	The credit utilization rate measures the percentage of available credit a consumer has used at a given point in time (in this case at the end of **June 2005**). It's calculated as follows: *Monthly balance / Credit Line.* Typically a higher utilization tends to have a negative impact on an overall 'FICO' type credit score and it may indicate a higher probability of	*if* LIMIT_BAL = 0 (*if* BILL_AMT4 > 0 *then* 1 *else 0*) *else* (*if BILL_AMT4> LIMIT_BAL* *then 1*	0 to 1

Feature name	Description and generic formula	Exact formula	How it's scaled
	default.	*else* BILL_AMT4 / LIMIT_BAL)	
UTIL_RT5	The credit utilization rate measures the percentage of available credit a consumer has used at a given point in time (in this case at the end of **May 2005**). It's calculated as follows: *Monthly balance / Credit Line.* Typically a higher utilization tends to have a negative impact on an overall 'FICO' type credit score and it may indicate a higher probability of default.	*if* LIMIT_BAL = 0 (*if* BILL_AMT5 > 0 *then* 1 *else 0*) *else* (*if BILL_AMT5> LIMIT_BAL* *then 1* *else* BILL_AMT5 / LIMIT_BAL)	0 to 1
UTIL_RT6	The credit utilization rate measures the percentage of available credit a consumer has used at a given point in time (in this case at the end of **April 2005**). It's calculated as follows: *Monthly balance / Credit Line.* Typically a higher utilization tends to have a negative impact on an overall 'FICO' type credit score and it may indicate a higher probability of default.	*if* LIMIT_BAL = 0 (*if* BILL_AMT6 > 0 *then* 1 *else 0*) *else* (*if BILL_AMT6> LIMIT_BAL* *then 1* *else* BILL_AMT6 / LIMIT_BAL)	0 to 1
PAY_RT1	The payment rate measures the amount of the credit balance the consumer has paid in full at a given point in time (in this case at the end of **September 2005**). This is calculated as follows: *Payment / Balance (caveat: a 0 payment to 0 balance ratio is considered 100% paid in full)*	*if* BILL_AMT1 = 0 OR PAY_AMT1 > BILL_AMT1 *then 1* *else* PAY_AMT1 / BILL_AMT1	0 to 1
PAY_RT2	The payment rate measures the amount of the credit balance the	*if* BILL_AMT2 = 0 OR PAY_AMT2 >	0 to 1

Feature name	Description and generic formula	Exact formula	How it's scaled
	consumer has paid in full at a given point in time (in this case at the end of **August 2005**). This is calculated as follows: *Payment / Balance (caveat: a 0 payment to 0 balance ratio is considered 100% paid in full)*	BILL_AMT2 *then 1* *else* PAY_AMT2 / BILL_AMT2	
PAY_RT3	The payment rate measures the amount of the credit balance the consumer has paid in full at a given point in time (in this case at the end of **July 2005**). This is calculated as follows: *Payment / Balance (caveat: a 0 payment to 0 balance ratio is considered 100% paid in full)*	*if* BILL_AMT3 = 0 OR PAY_AMT3 > BILL_AMT2 *then 1* *else* PAY_AMT3 / BILL_AMT3	0 to 1
PAY_RT4	The payment rate measures the amount of the credit balance the consumer has paid in full at a given point in time (in this case at the end of **June 2005**). This is calculated as follows: *Payment / Balance (caveat: a 0 payment to 0 balance ratio is considered 100% paid in full)*	*if* BILL_AMT4 = 0 OR PAY_AMT4 > BILL_AMT4 *then 1* *else* PAY_AMT4 / BILL_AMT4	0 to 1
PAY_RT5	The payment rate measures the amount of the credit balance the consumer has paid in full at a given point in time (in this case at the end of **May 2005**). This is calculated as follows: *Payment / Balance (caveat: a 0 payment to 0 balance ratio is considered 100% paid in full)*	*if* BILL_AMT5 = 0 OR PAY_AMT5 > BILL_AMT5 *then 1* *else* PAY_AMT5 / BILL_AMT5	0 to 1
PAY_RT6	The payment rate measures the amount of the credit balance the consumer has paid in full at a given point in time (in this case at the end of **April 2005**). This is calculated as follows: *Payment / Balance (caveat: a 0 payment to 0 balance ratio is considered 100% paid in full)*	*if* BILL_AMT6 = 0 OR PAY_AMT6 > BILL_AMT6 *then 1* *else* PAY_AMT6 / BILL_AMT6	0 to 1
UTIL_VEL1	The utilization velocity measures how fast the utilization rate	UTIL_RT1 - UTIL_RT2	-1 to 1

Feature name	Description and generic formula	Exact formula	How it's scaled
	increased or decreased from last month (in this case from **August 2005 to September 2005**).		
UTIL_VEL2	The utilization velocity measures how fast the utilization rate increased or decreased from last month (in this case from July 2005 to **August 2005**).	UTIL_RT2 - UTIL_RT3	-1 to 1
UTIL_VEL3	The utilization velocity measures how fast the utilization rate increased or decreased from last month (in this case from **June 2005 to July 2005**).	UTIL_RT3 - UTIL_RT4	-1 to 1
UTIL_VEL4	The utilization velocity measures how fast the utilization rate increased or decreased from last month (in this case from **May 2005 to June 2005**).	UTIL_RT4 - UTIL_RT5	-1 to 1
UTIL_VEL5	The utilization velocity measures how fast the utilization rate increased or decreased from last month (in this case from **April 2005 to May 2005**).	UTIL_RT5 - UTIL_RT6	-1 to 1
TOT_UTIL_VEL	The total utilization velocity measures the overall (average) increase or decrease in utilization over the six month history for each customer.	(UTIL_VEL1 + UTIL_VEL2 + UTIL_VEL3 + UTIL_VEL4 + UTIL_VEL5) / 5	-1 to 1
UTIL_DIFF1	The difference between the minimum utilization rate and the current utilization rate (for **September 2005**) for a given customer over the six month history shows if the customer has radically increased his/her utilization which in turn may be an indicator for default.	UTIL_RT1 – MIN (UTIL_RTL1, UTIL_RT2 , UTIL_RT3 , UTIL_RT4 , UTIL_RT5, UTIL_RT6)	0 to 1
UTIL_DIFF2	The difference between the minimum utilization rate and the current utilization rate (for **August 2005**) for a given customer over the six month history shows if the customer has radically increased his/her utilization which in turn	UTIL_RT2 – MIN (UTIL_RTL1, UTIL_RT2 , UTIL_RT3 , UTIL_RT4 , UTIL_RT5, UTIL_RT6)	0 to 1

Feature name	Description and generic formula	Exact formula	How it's scaled
	may be an indicator for default.		
UTIL_DIFF3	The difference between the minimum utilization rate and the current utilization rate (for **July 2005**) for a given customer over the six month history shows if the customer has radically increased his/her utilization which in turn may be an indicator for default.	UTIL_RT3 – MIN (UTIL_RTL1, UTIL_RT2 , UTIL_RT3 , UTIL_RT4 , UTIL_RT5, UTIL_RT6)	0 to 1
UTIL_DIFF4	The difference between the minimum utilization rate and the current utilization rate (for **June 2005**) for a given customer over the six month history shows if the customer has radically increased his/her utilization which in turn may be an indicator for default.	UTIL_RT4 – MIN (UTIL_RTL1, UTIL_RT2 , UTIL_RT3 , UTIL_RT4 , UTIL_RT5, UTIL_RT6)	0 to 1
UTIL_DIFF5	The difference between the minimum utilization rate and the current utilization rate (for **May 2005**) for a given customer over the six month history shows if the customer has radically increased his/her utilization which in turn may be an indicator for default.	UTIL_RT5 – MIN (UTIL_RTL1, UTIL_RT2 , UTIL_RT3 , UTIL_RT4 , UTIL_RT5, UTIL_RT6)	0 to 1
UTIL_DIFF6	The difference between the minimum utilization rate and the current utilization rate (for **April 2005**) for a given customer over the six month history shows if the customer has radically increased his/her utilization which in turn may be an indicator for default.	UTIL_RT6 – MIN (UTIL_RTL1, UTIL_RT2 , UTIL_RT3 , UTIL_RT4 , UTIL_RT5, UTIL_RT6)	0 to 1
TOT_PAY_RT	The total payment rate measures the overall (average) rate at which the customer the consumer has paid in full the amount of credit balance he/ she is liable for over the six month history.	(PAY_RT1 + PAY_RT2 + PAY_RT3 + PAY_RT4 + PAY_RT5 + PAY_RT6) / 6	0 to 1
AGE_60_ABOVE	Customers who are 60 years old and above	AGE >= 60	Not applicable
AGE_45_59	Customers between the ages of 45 to 59	45<= AGE < 60	Not applicable
AGE_40_44	Customers between the ages of 40 to 44	40 <= AGE < 45	Not applicable

Feature name	Description and generic formula	Exact formula	How it's scaled
AGE_39_39	Customers between the ages of 36 to 39	36 <= AGE < 40	Not applicable
AGE_33_35	Customers between the ages of 33 to 35	33 <= AGE < 36	Not applicable
AGE_30_32	Customers between the ages of 30 to 32	30 <= AGE < 33	Not applicable
AGE_29_BELOW	Customers who are 29 years old and younger	AGE < 30	Not applicable

The binning criteria for the age variable is derived use some statistical analysis of the data. The first statistical graph below in figure 3.1 shows that age is almost identically distributed amongst defaulters and non-defaulters alike: the mean or median is around 35, the bulk of customers (interquartile range) is roughly between the ages of 28 to 43 and those who are above 60 are just outliers (there aren't a lot of 60+ year old customers in the data set). This allows us to derive the following 4 initial age bins:

- Those above 60 and above (the outliers)
- Those between 41 and 60 (above 3rd interquartile mark)
- Those between 28 to 41 (interquartile range)
- Those below 280 (below1st interquartile mark)

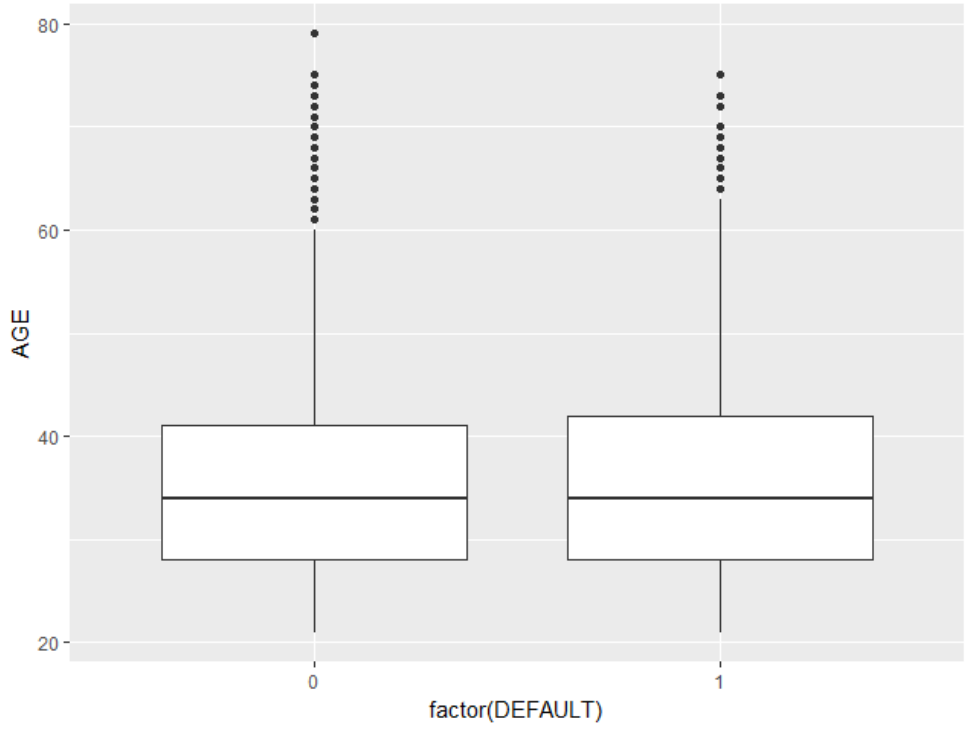

Figure 3.1: *Boxplot of age classified by default (no = 0, yes = 1)*

These initial bins can be broken down further – specifically the interquartile group that 28 to 41 which contains the majority of customers can be broken down further. The second graph in figure 3.2 is a decision tree (a method of looking at primary features that might provide insight and then splitting it along some critical decision boundary line) that among other things shows that customers who are generally older than 35 are more likely to married. Married customers who are 39 or older are more likely to have high degree of education while single customers who are highly educated tend to be between 30 to 32. Single customers who are younger than 30 tend to carry a lower credit limit. Female customers who are highly educated and married tend be younger than 39 while male customers who are highly educated and married tend to be older than 39. Using this insight it is possible to break down the 28 to 32 group further to reflect their

correlation strength with other variables like gender, marital status, education level and credit limit:

- 39 to 43 (instead of 41): married and highly educated
- 35 to 39: married and highly education and female
- 32 to 35: single
- 30 to 32: single and highly educated
- Younger than 30: the rest

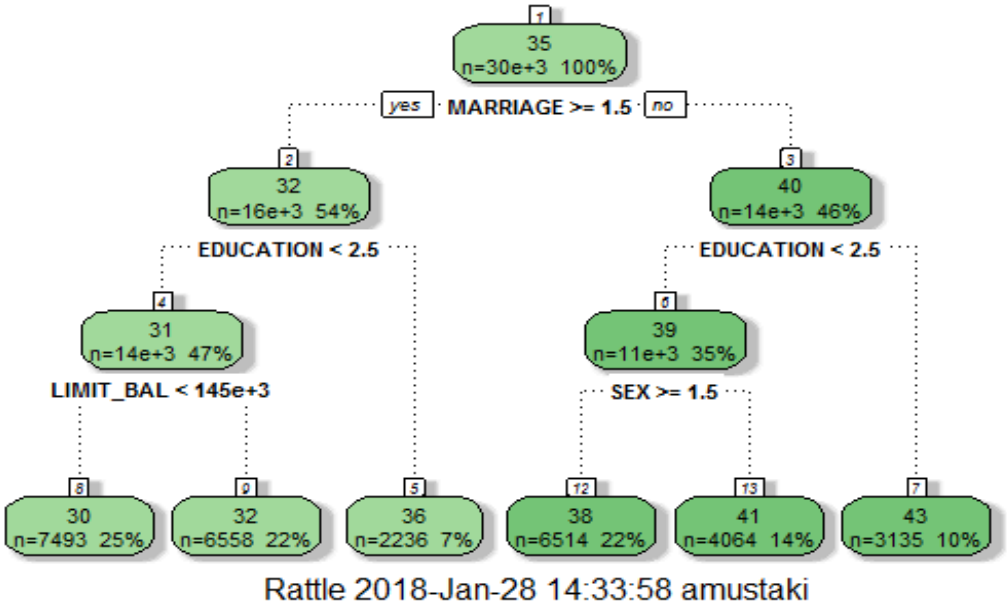

Figure 3.2: *Age decision tree*

Thus the final age bins are as follows:

Age bin	Description
60 and above	Outliers
45 to 59	Above 3rd interquartile mark
40 to 44	Married and highly educated
36 to 39	Married and highly education and female
33 to 35	Single
30 to 32	Single and highly educated
29 and younger	The rest

4. Exploratory Data Analysis

4a. Traditional EDA

From figure 4.1 below, it looks like the vast majority of the customers in the data set are between 20 and 60 years of age and they have a credit limit under 500,000 New Taiwanese Dollar (approximately $17,000 USD), which for the most part they do not seem to exceed in their monthly spending.

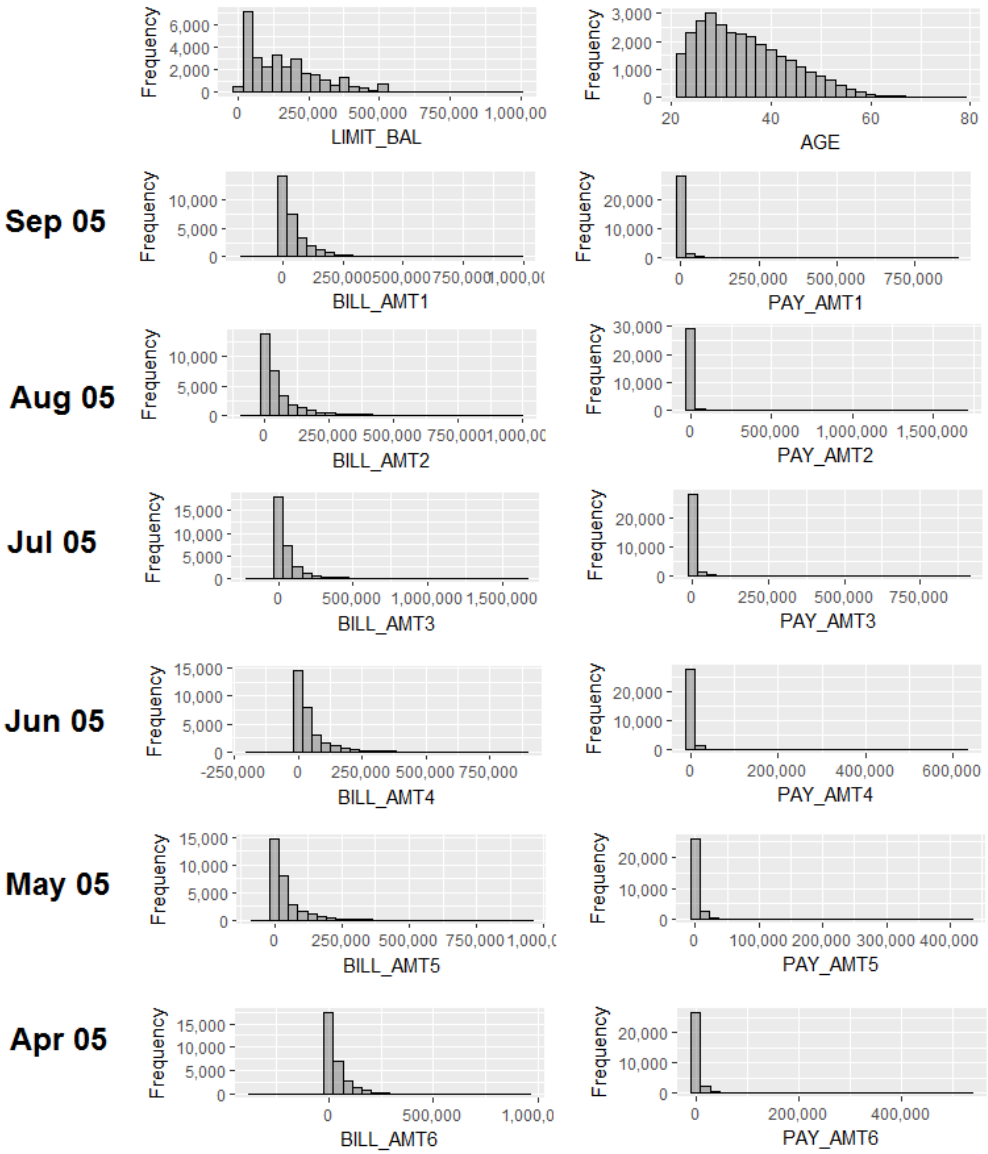

Figure 4.1 Histograms of continuous variables

Figure 4.2 below shows that there are more female customers than male; most customers have a university or a graduate degree; and there are slightly more single customers than married ones. It also shows that most customers did not default on their payments, as such the breakdown of defaulting vs non-defaulting customers by age, gender, education level and marital status is inconclusive (colorful bar charts on the right). It also shows that most customers are usually on time or at most one month late with their payments. However a look at the missing data in figure 4.3 below shows that vast majority of the late payment indicator fields (i.e. PAY_*X)* have missing data and therefore these fields are not very useful in the forthcoming modelling exercise.

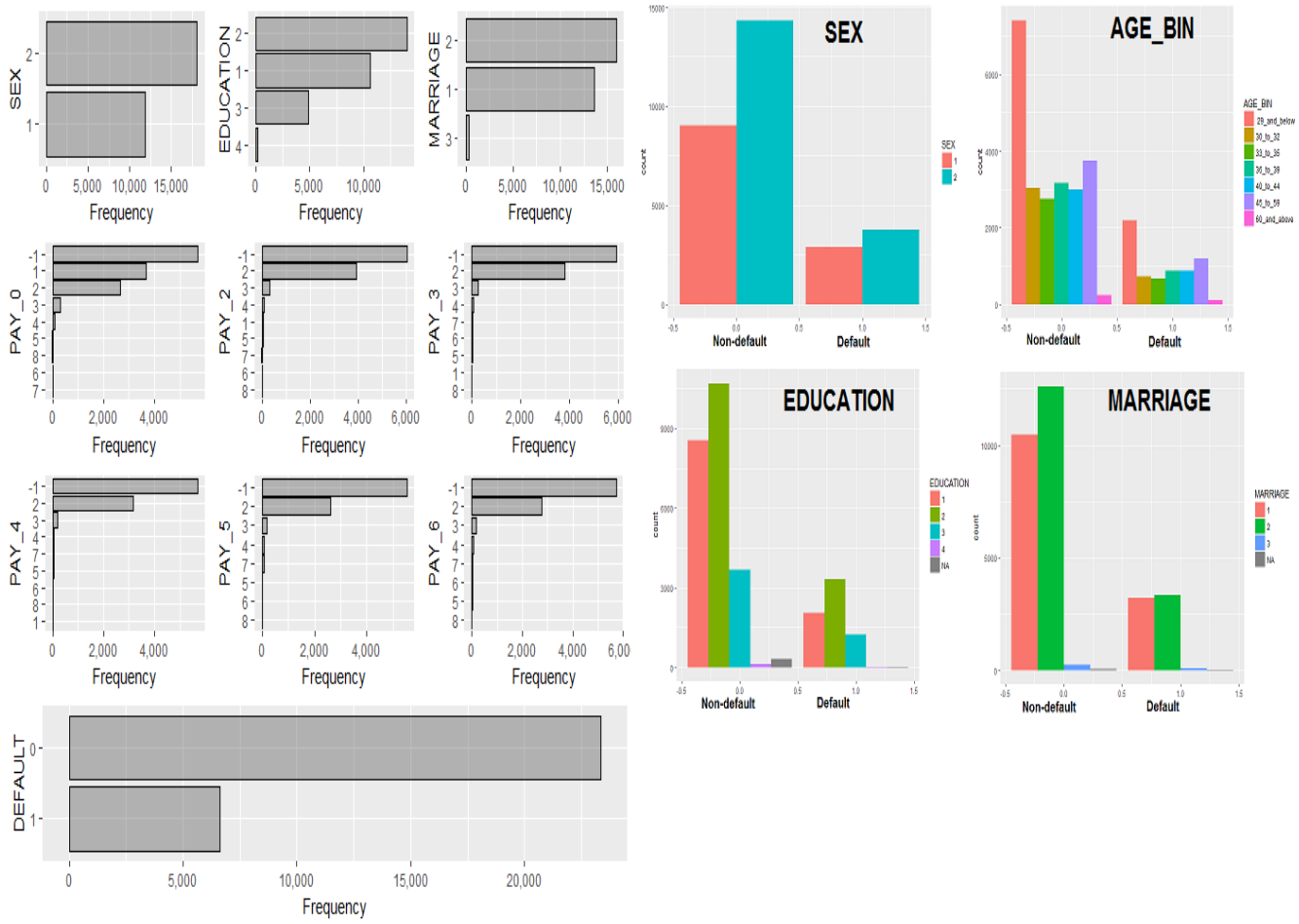

Figure 4.2 Boxplots of discreet variables

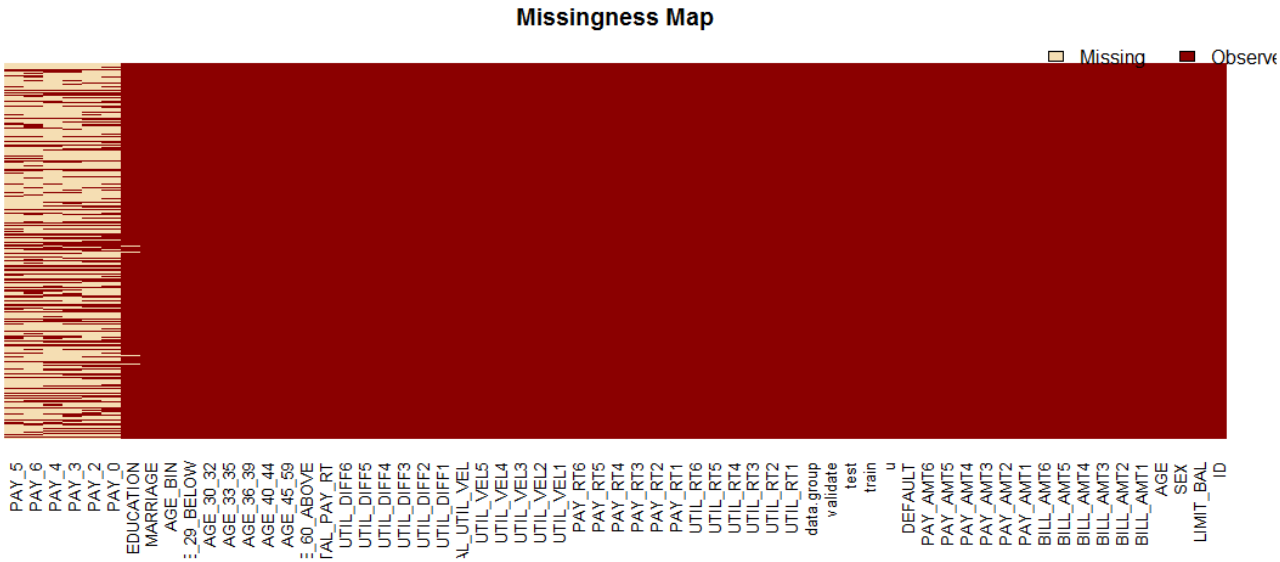

Figure 4.3 Missing data

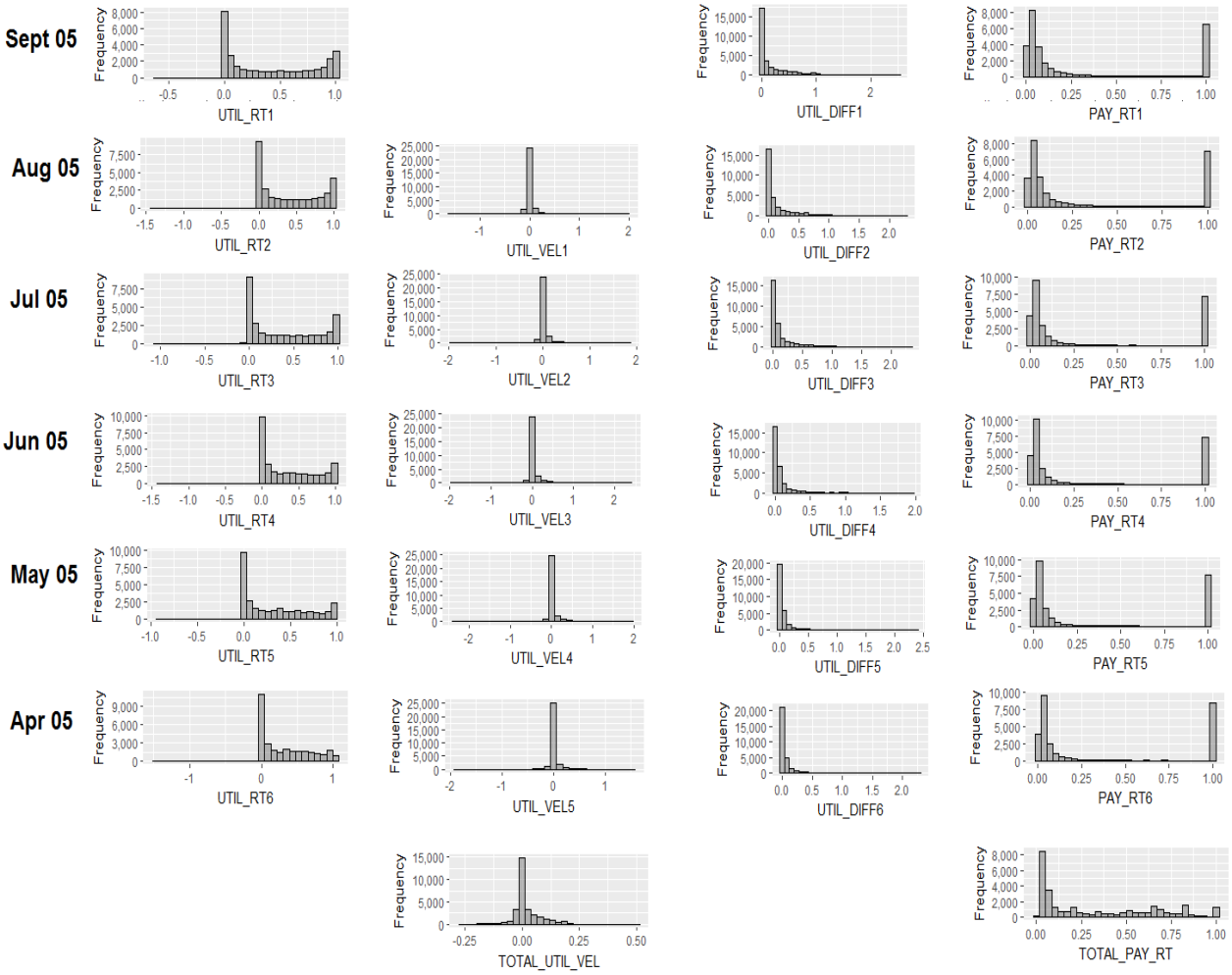

Figure 4.4 Continuous Engineered Features - Histograms

In figure 4.4 above we see that all of the engineered features which are continuous (which are all the engineered features except for the age bins) fall within their expected scale:

Engineered feature	Scale	Concur with EDA plot above
Utilization rate UTIL_RT*X*	0 to 1	Yes
Utilization difference from the minimum UTIL_DIFF*X*	0 to 1	Yes
Payment rate PAY_RT*X*	0 to 1	Yes
Total payment rate TOTAL_PAY_RT	0 to 1	Yes
Utilization velocity UTIL_VEL*X*	-1 to 1	Yes
Total utilization velocity TOTAL_UTIL_VEL *(Note: Since this is an average most values fall between -0.25 and 0.5)*	-1 to 1	Yes

Table 4.1: *Data summaries in lieu data quality check for engineered features*

In terms of the shapes of the histograms in figure 4.4:

- The utilization rate (UTIL_RT*X*) histograms are slightly right-skewed with peaks at 0 and 1 indicating that the largest group of customers do not fully utilize their full credit limit and the second largest group of customers do.
- The utilization velocities (UTIL_VEL*X*) and total utilization velocity (TOTAL_UTIL_VEL) histograms are unimodal and mostly centered around 0, which seems to indicate that most customers do not make a habit of radically increasing or decreasing their utilization from month to month – i.e. in general utilization rates tend to be the same from month to month.
- The histograms for the difference between the monthly utilization rate and the minimum utilization rate (UTIL_DIFF*X*) which shows if the customer has radically increased

his/her utilization which in turn may be an indicator for default, are right skewed and peaks around 0. This concurs with the earlier assessment that most customers do not radically alter their spending habits from month to month.

- The histograms for the payment rates (PAY_RT*X*) are right skewed with peaks at 0 and 1. The histogram for the total payment rate (TOTAL_PAY_RT) does not have a peak at 1. This seems to indicate that a lot of customer pay their bill in full but most do not.

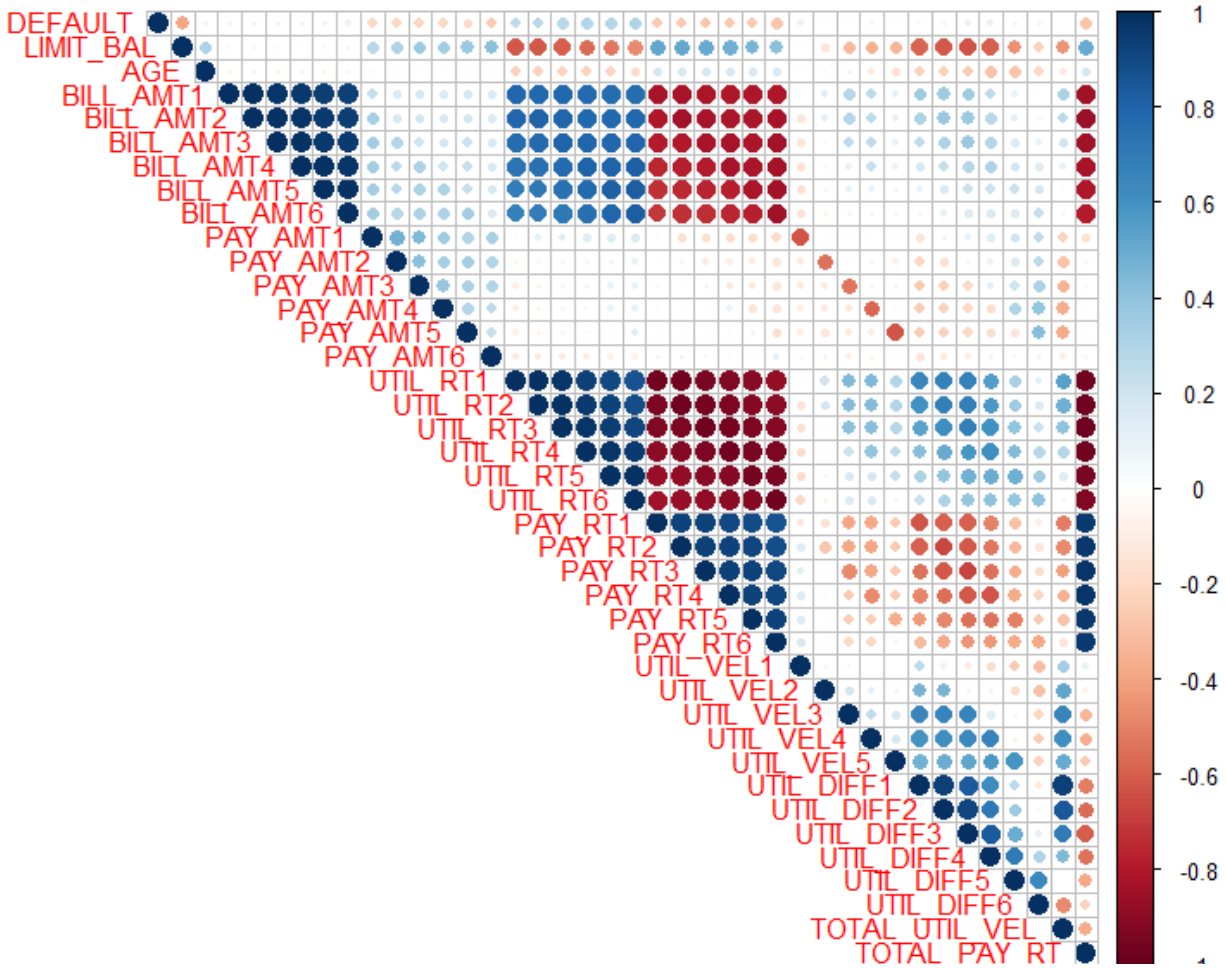

Figure 4.5 Correlation plot of continuous variables (including most engineered features)

The correlation plot in figure 4.5 above shows that the default indicator is somewhat positively correlated with the monthly utilization rates and negatively correlated with large

payment amounts, payment rates and the total payment rate. Amongst the engineered features the following correlations are observed:

- the monthly billed amount (BILL_AMT*X*) variables are positively correlated with utilization rates and negatively correlated with payment rates and total payment rate
- monthly utilization rates are positively correlated with the monthly utilization difference from the minimum and negatively correlated with payment rates and total payment rate
- the monthly payment rates are negatively correlated with the monthly utilization differences from the minimum
- the monthly utilization velocities are positively correlated with the monthly utilization differences from the minimum
- total utilization velocity is most positively correlated with the monthly utilization differences from the minimum
- total payment rates are most negatively correlated with the utilization rates

These correlations are quite intuitive and in fact figure 4.6 shows that:

- customers who defaulted on their payments had a slightly average higher utilization rate (0.5) than customers who did not default
- customers who had 6 month higher average payment rate at around 0.25 did not default on their payments; customers who default had an average payment rate of near 0
- the utilization velocity and highest utilization from minimum was not that different between defaulters and non-defaulters

Therefore a preliminary assessment can be made that utilization rate and total payment rate may have higher predictive ability of default than the other engineered features.

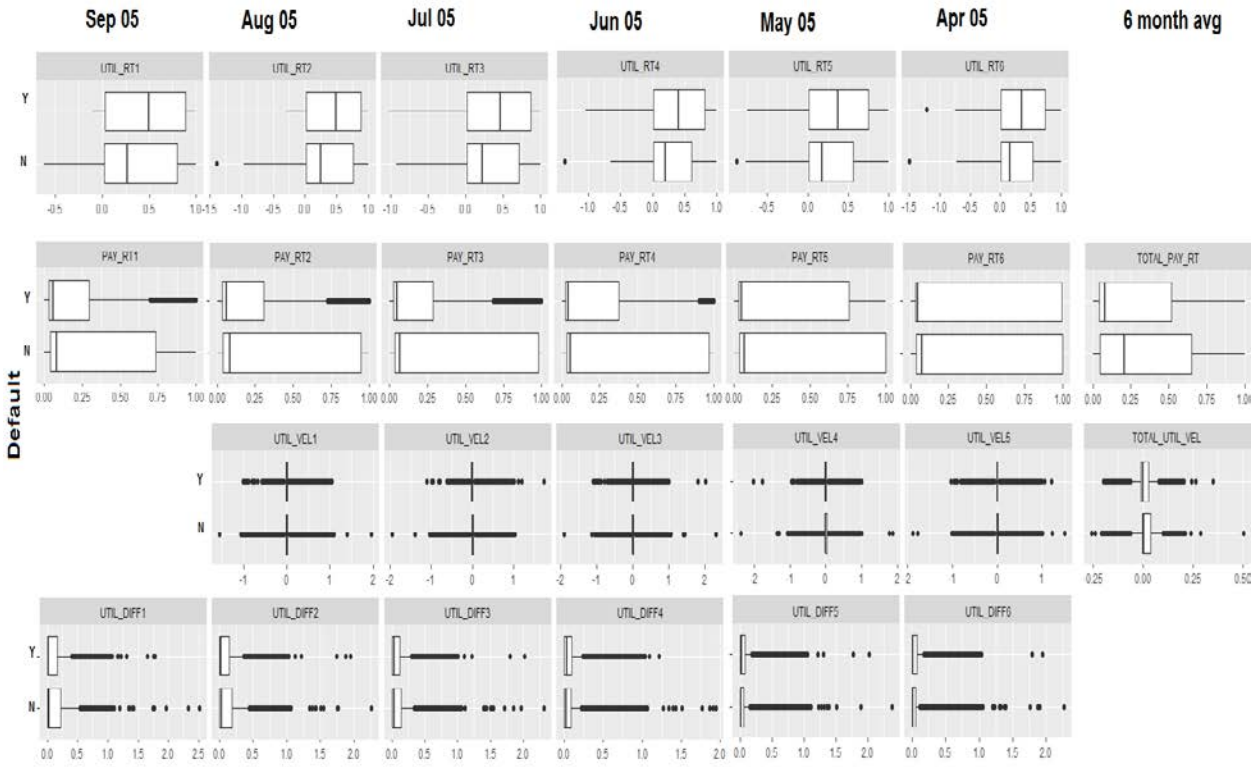

Figure 4.6 Box plots of engineered variables against default indicator

As far as the discreet variables in the data set are concerned, a closer examination of the socio-demographics attributes (gender, education level, marital status and age) and their relationship with defaulting on credit card payments show that defaulters in general are allocated a slightly lower credit limit across all respective age groups, education levels, marital statuses and gender (figures 4.7, 4.8, 4.9 and 4.10 and below). Also female customers tend to have a slight higher credit limit than their male counterparts in both the default and non-default segments.

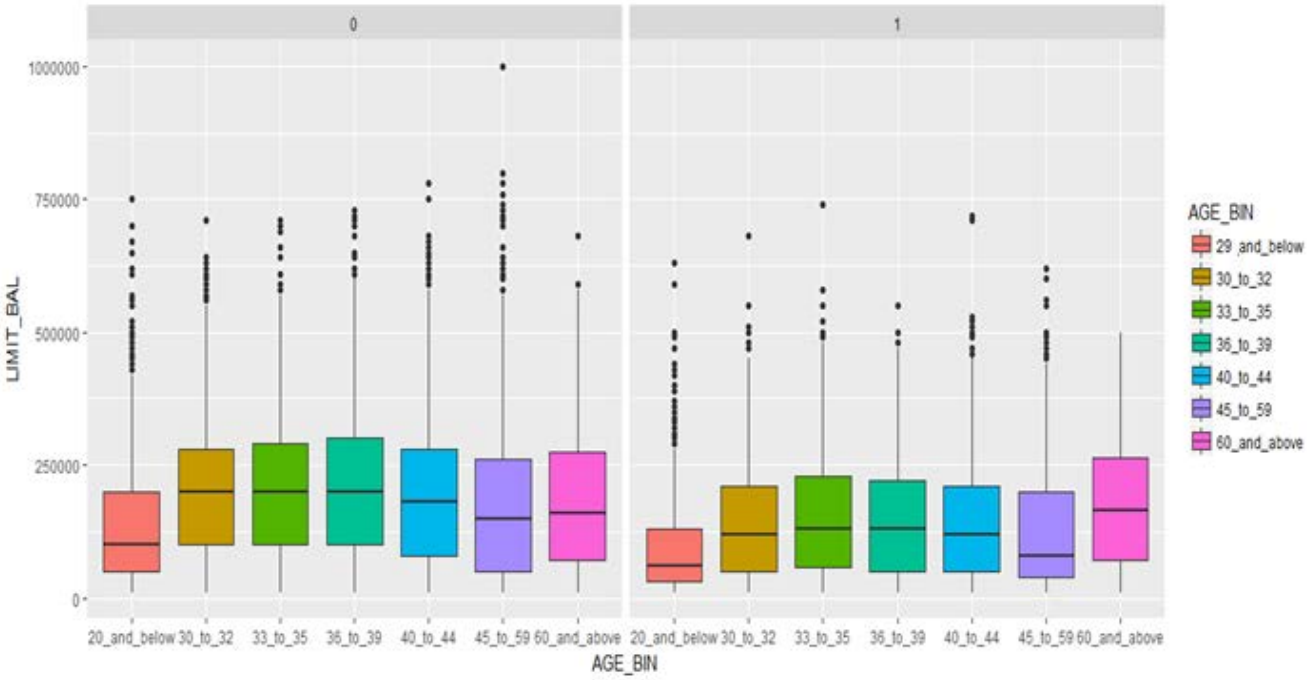

Figure 4.7: *Credit limit allocated to different age groups, for defaulters (in column marked 1 on the right) and non-defaulters (column marked 0 on the left)*

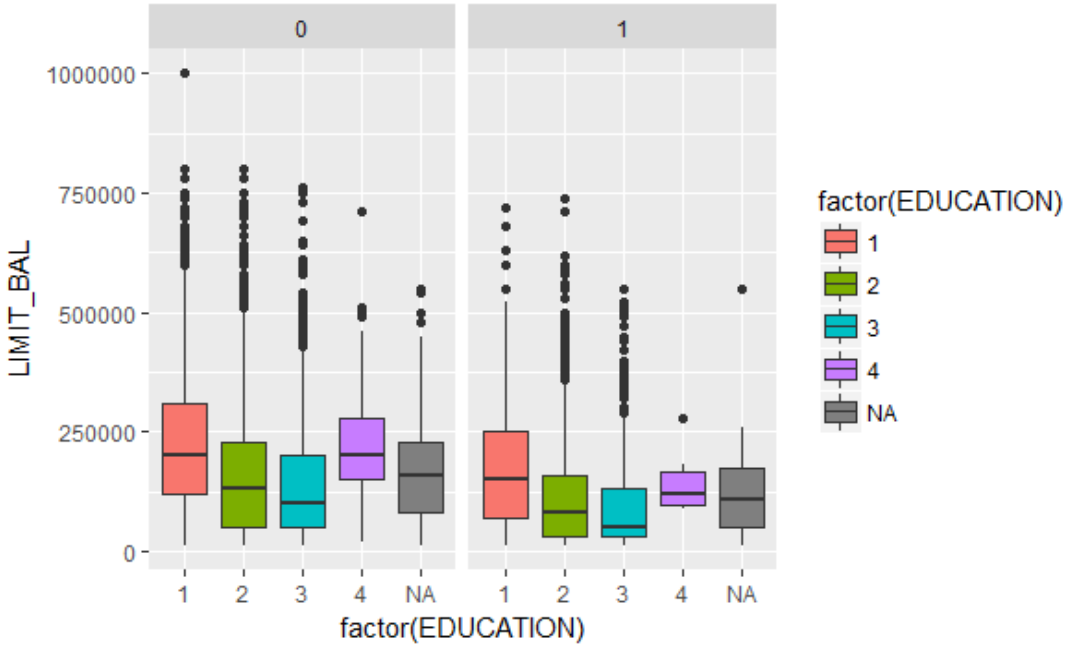

Figure 4.8: *Credit limit allocated to customers with different levels of education, for defaulters (in column marked 1 on the right) and non-defaulters (column marked 0 on the left)*

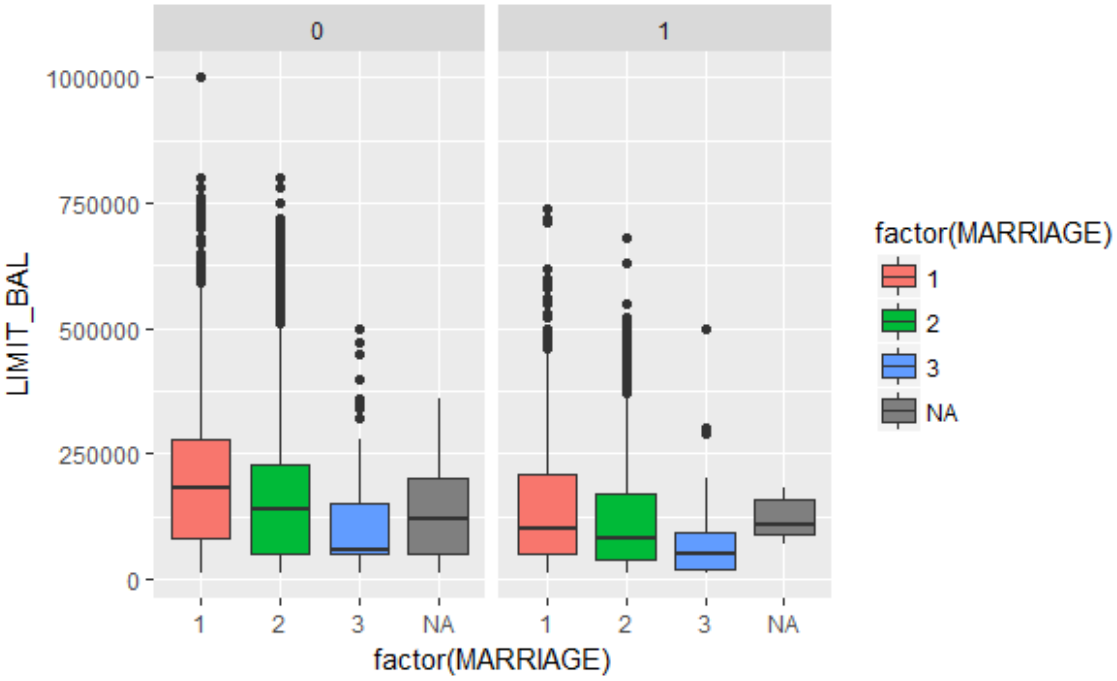

Figure 4.9: *Credit limit allocated to customers with different marital statuses, for defaulters (in column marked 1 on the right) and non-defaulters (column marked 0 on the left)*

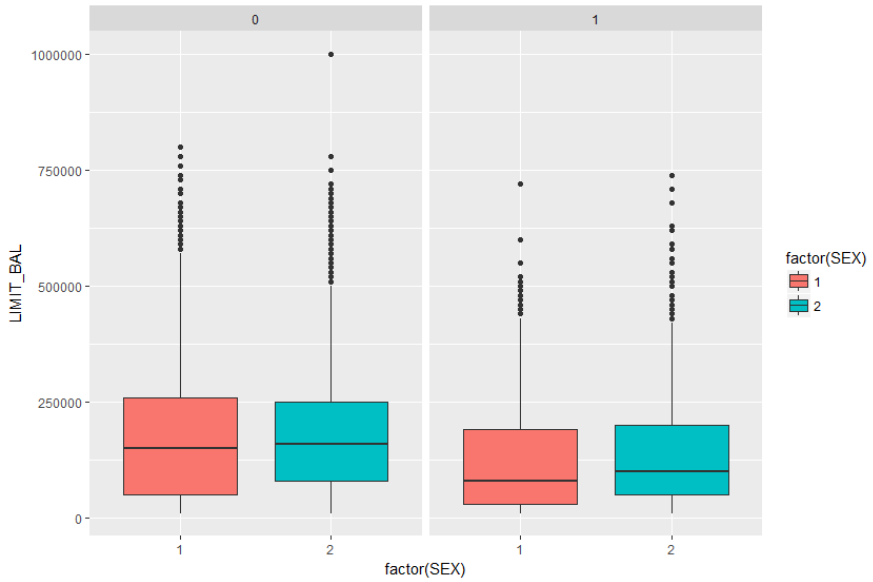

Figure 4.10: *Credit limit allocated to male (SEX=1) and female (SEX = 2) customers, who were both defaulters (in column marked 1 on the right) and non-defaulters (column marked 0 on the left)*

4b. Model Based EDA

So far we have seen that defaulters in general have:

- a lower credit limit across all age groups, genders, marital statuses and education levels
- an average utilization rate around 0.5
- a 6 month average payment rate almost near 0

In order to explore the relationship between credit limit, utilization rates, 6 month average payment rate and default, the following decision trees (figures 4.11, 4.12 and 4.13) are employed to discern any observable pattern:

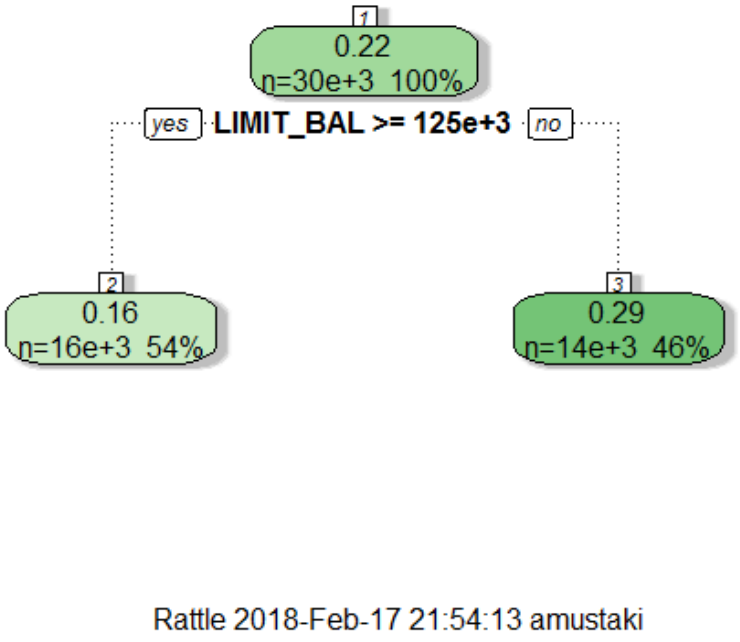

Figure 4.11: *Decision tree: Defaulters in general have a credit limit below 125,000 NTD*

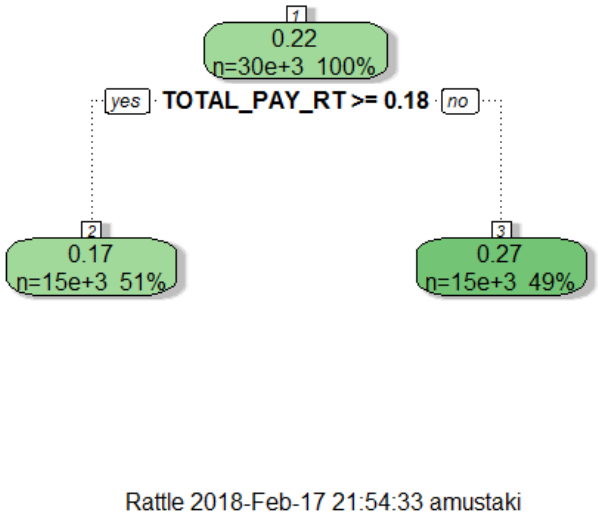

Figure 4.12: *Decision tree: Defaulters in general have a 6 month average payment rate below 0.18*

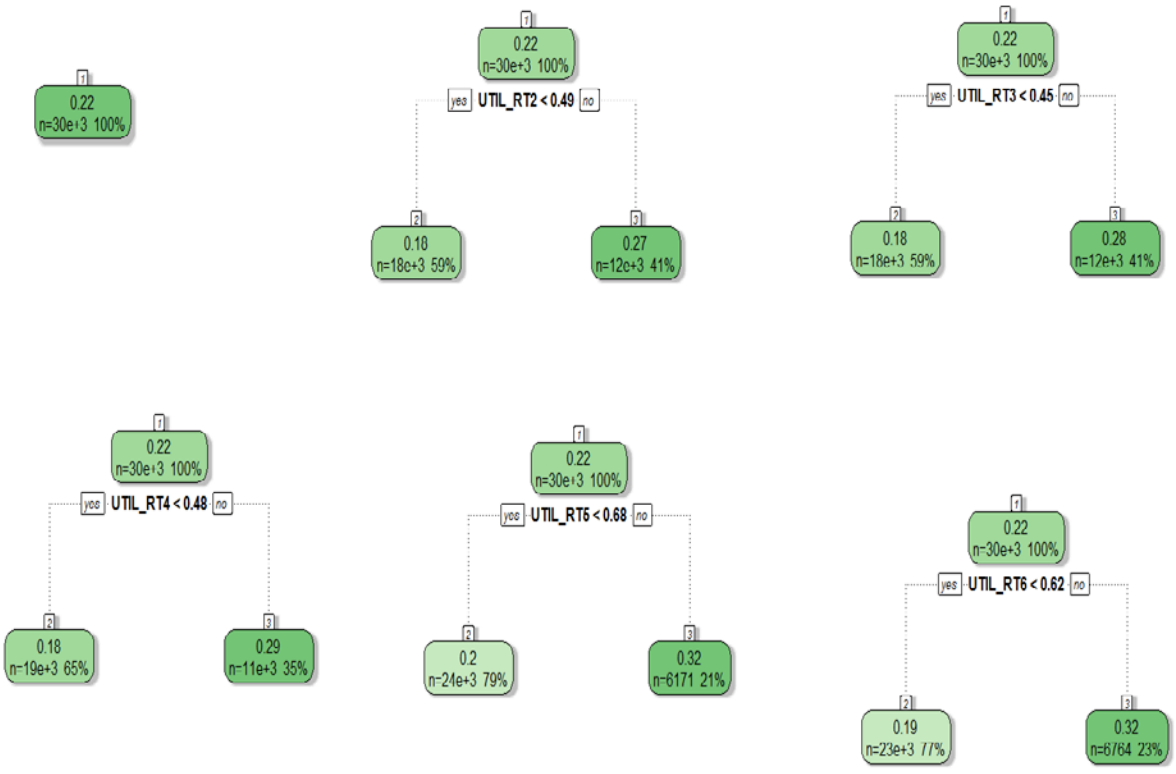

Figure 4.13: *6 Decision trees: Defaulters in general have a UTIL_RT2 above 0.49, UTIL_RT3 above 0.45, UTIL_RT4 above 0.48, UTIL_RT5 above 0.68 and UTIL_RT6 above 0.62*

Based on the decision trees above the following data summaries and class assessments can be made and their relationship with the default indicator can be investigated further:

Engineered feature	Class Boundary	Conceptual relationship with default
UTIL_RT2	0.49	Values equal to or above indicate default
UTIL_RT3	0.45	Values equal to or above indicate default
UTIL_RT4	0.48	Values equal to or above indicate default
UTIL_RT5	0.68	Values equal to or above indicate default
UTIL_RT6	0.62	Values equal to or above indicate default
TOTAL_PAY_RT	0.18	Values below indicate default
LIMIT_BAL	125,000	Values below indicate default

Table 4.2: *Data summaries from decision trees: class boundaries for utilization rates, 6 month average payment rates and credit limit to determine relationship with default*

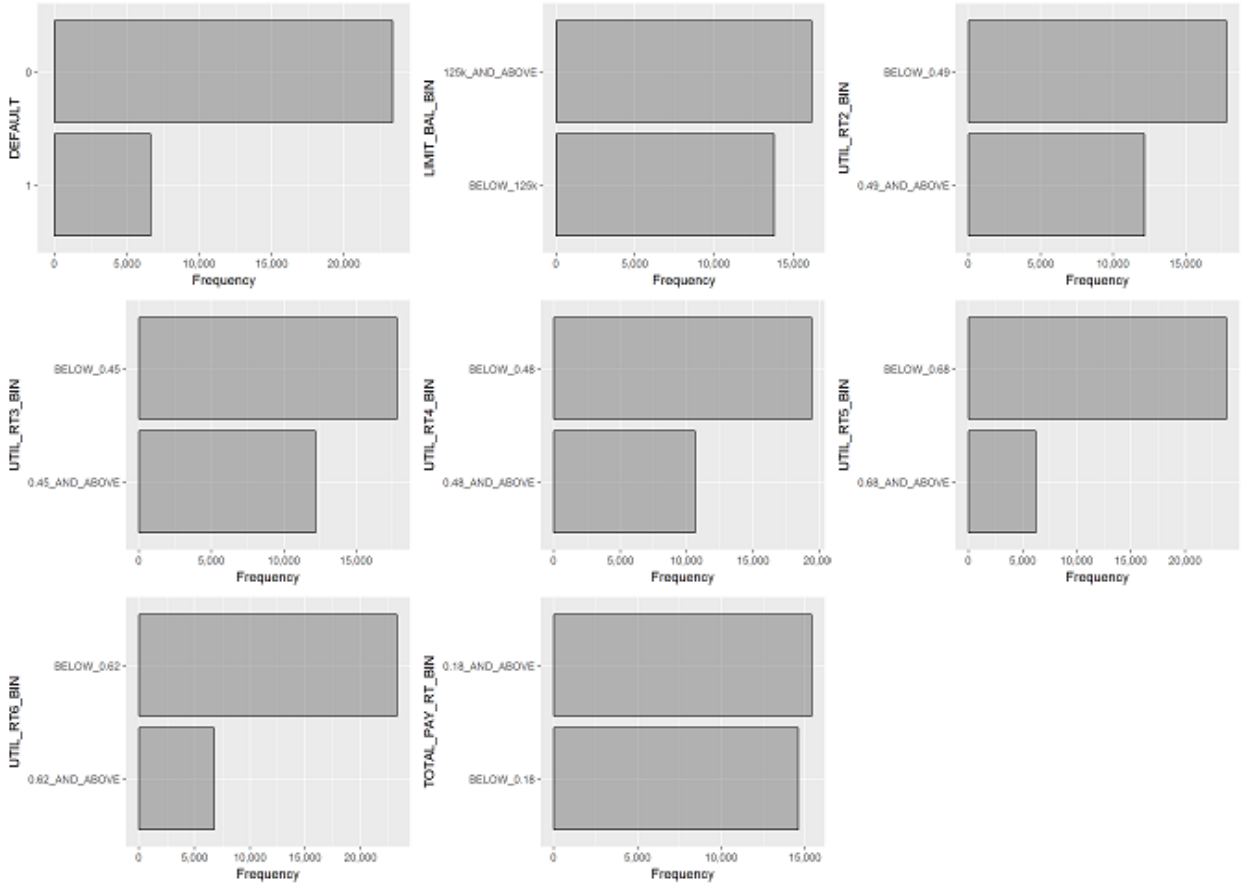

Figure 4.14: *Bar plots of newly created class boundary variables*

Withstanding the fact that there are more non-defaulters than there are defaulters in the data set, figure 4.14 shows that the data set is more or less even split on:

- LIMIT_BAL above and below 125,000 NTD
- TOTAL_PAYMENT_RT above and below 0.18

However when it comes to utilization rates the data is not as evenly distributed as there are more customers in the data set with:

- UTIL_RT2 below 0.49
- UTIL_RT3 below 0.45
- UTIL_RT4 below 0.48
- UTIL_RT5 below 0.68
- UTIL_RT6 below 0.62

This somewhat echoes the uneven distribution of more non-defaulters vs. less defaulters in the data set as seen in figure 4.15 below. So it is unclear whether these class boundaries yield any actual predictive power or if they simply coincide with the distribution of non-defaulters vs. non-defaulters in the data set. Either way, when a predictive model is constructed, a general assumption can be made that:

- higher utilization rates tend to coincide with higher default rates
- lower average payment rates and lower credit limit balances tend to coincide with higher default rates

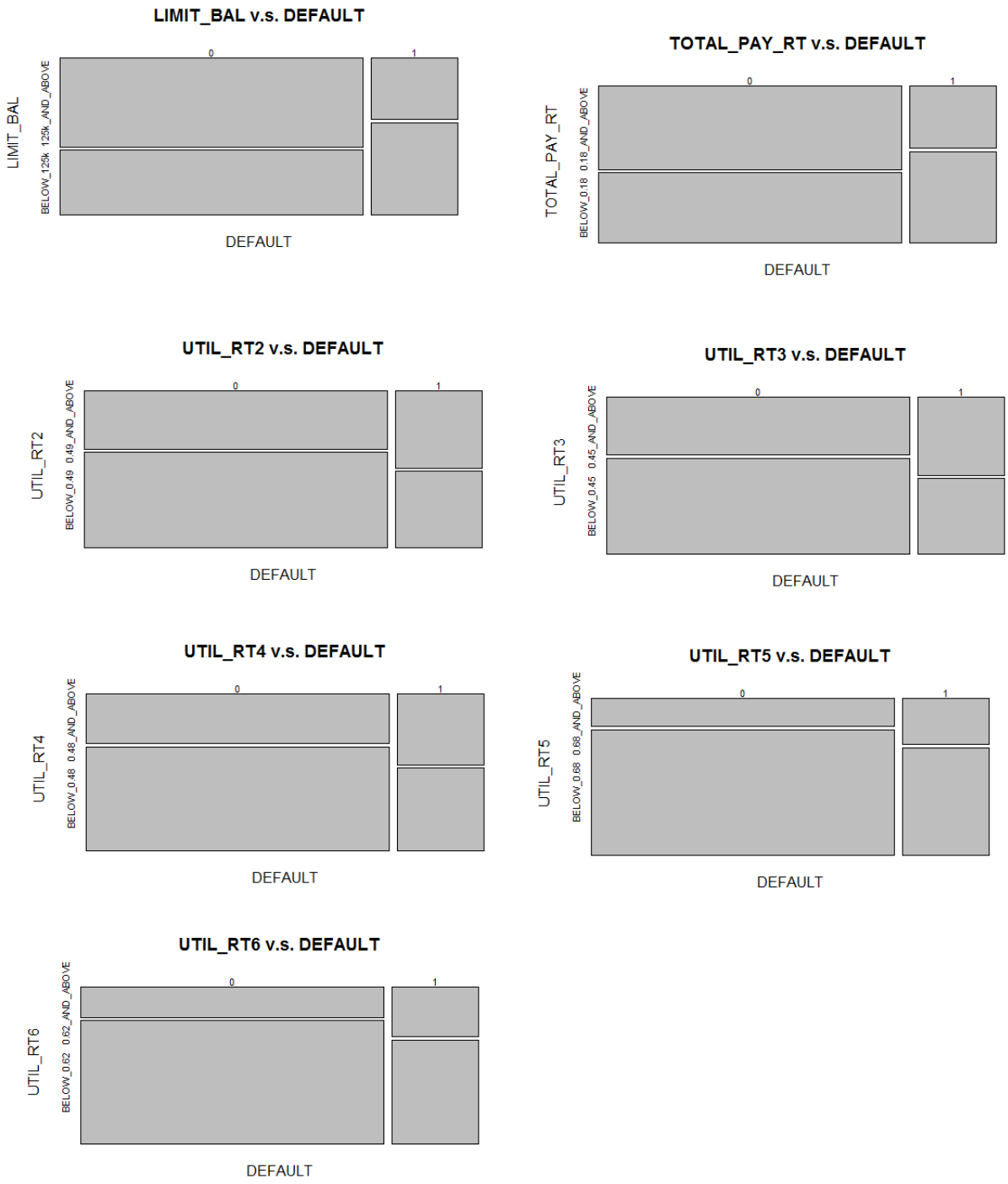

Figure 4.15: *Mosaic plots of class boundaries vs. default*

5. Predictive Modeling: Methods and Results

To predict the probability of default using all the variables discussed so far, these 4 different modeling approaches are taken:

1. **Random Forrest** (one of the top modelling approaches used by Kaggle data science competition winners and it uses a lot of or a "forest" of different decision trees and then takes the average response of all the trees to arrive at an optimal answer)
2. **Gradient Boosting** (a more optimized version of random forest which requires more fine-tuning and it basically bootstrap aggregates or "bags" models (i.e. reduces variance in prediction by generating additional training data from original dataset) while boosting model performance iteratively by combining different models with a an eye towards reducing error)
3. **Logistic Regression with Variable Selection** (similar to regular regression except the output is categorical, in this case binary ("defaulted" or "did-not-default") and a subset of the predictor variables are used to keep the model simple and optimal)
4. **Support Vector Machine** (a supervised machine learning technique that is based on the idea of finding a hyperplane that divides the dataset into 2 classes)

For each of these models an appropriate level of model output along with a table of the model performance in-sample (i.e. on the training data set) and out-of-sample (i.e. on the test data set) are presented below. These performance metrics are derived using the confusion matrix structure shown below and their formulae is provided in table 5.1 below:

	Predicted Non-Default	Predicted Default
Actual Non-Default	True Negative	False Positive
Actual Default	False Negative	True Positive

Performance Metric	Formula
True Positive Rate (Sensitivity)	Number of true positive cases / (Number of true positive case + Number of false negative cases)
False positive rate	Number of false positive cases / (Number of false positive + Number of true negatives)
Model accuracy (% of records correctly classified)	(Number of true negative cases + Number of true positive cases) / Total number of cases

Table 5.1: *Confusion Matrix Structure & Performance Metric Formulae*

5.a Random Forest (RF)

First, a random forest model is generated to solve the classification problem at hand: predict the probability of default using the available variables using 500 trees and 8 different variables at each split. Figure 5.1 below shows a plot of the error vs. the number trees, which shows the number of errors encountered for the different classes (0 in red for non-default and 1 in green for default) and out-of-bag or OOB samples (in black) over the amount of trees. The OOB error is a method for estimating the prediction error, and it is calculated "using observations not trained on for each decision tree in the forest and aggregates over all so there should be no bias, hence the name out-of-bag" (*Colakoglu et. al. [1]*). The OOB error is 21% which means there is a 79% out of sample accuracy for the training set (i.e. the model accuracy formula shown in table 5.1 above).

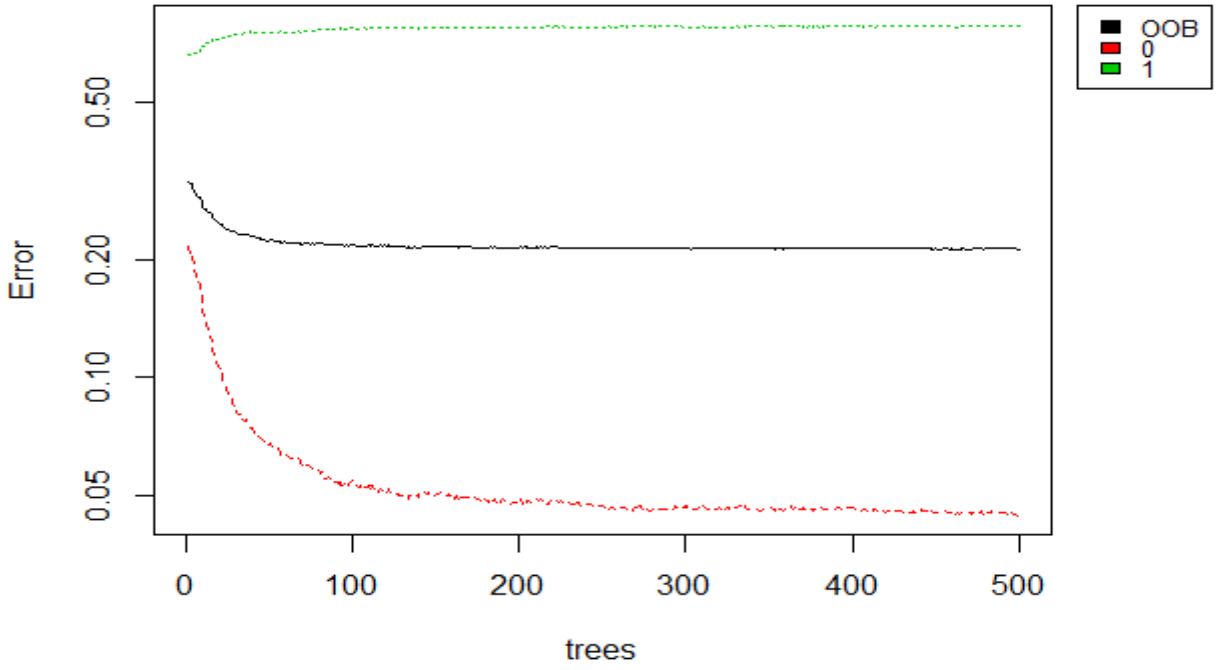

Figure 5.1: *Error vs. Number of trees (OOB = Out Of Bound, 0 = Non-Default, 1 = Default)*

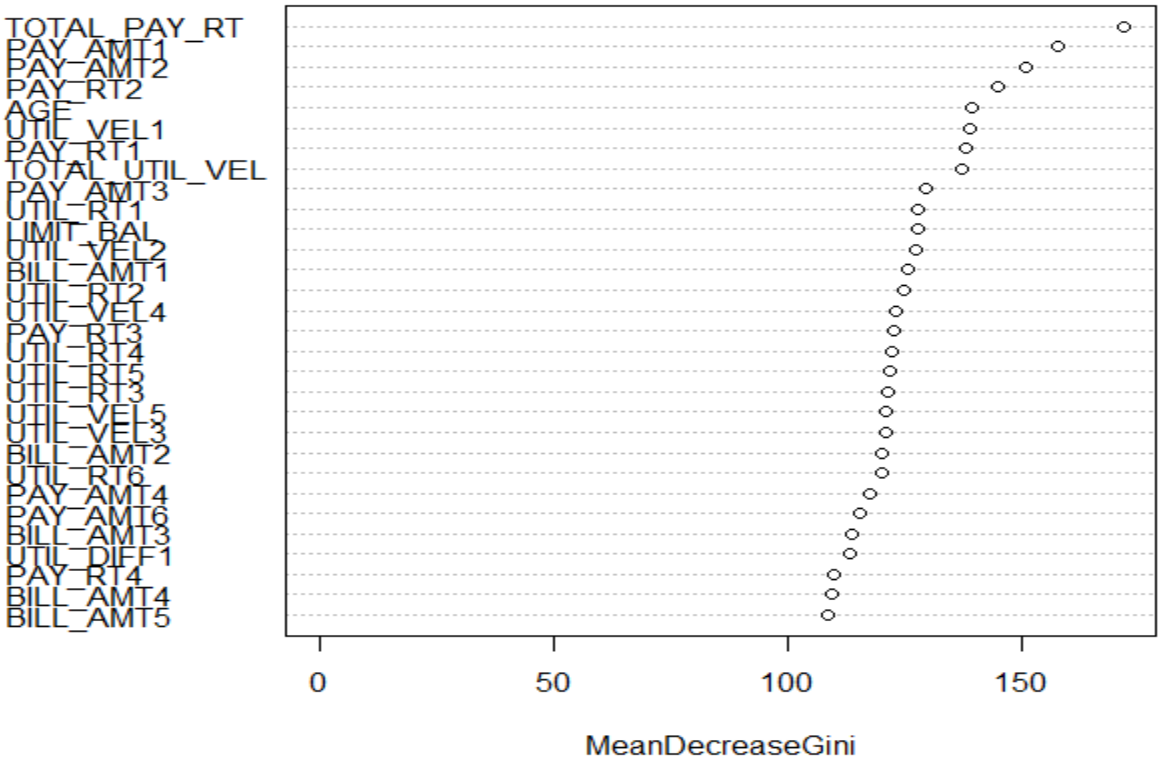

Figure 5.2: *Variable Importance Plot – Random Forrest*

Figure 5.2 above shows the variable importance plot, which shows the most important variables in the random forest model. The x axis represents the mean decrease in node impurity as different variables are introduced in the model. We notice that the total payment rate has the highest impact on the model, followed by the payment amounts for the last 2 months in the data set. Age, some of the payment rates, utilization rates, utilization velocities, the monthly billed amount were some of the other variables that were considered. Table 5.2 and 5.3 below shows the confusion matrix for the training and test data sets:

	Predicted Non-Default	Predicted Default
Actual Non-Default	11058	517
Actual Default	2661	746

Table 5.2: *Confusion Matrix for Random Forrest (training data)*

	Predicted Non-Default	Predicted Default
Actual Non-Default	5431	1255
Actual Default	243	293

Table 5.3: *Confusion Matrix for Random Forrest (test data)*

Here is the model performance summary for the random forest model:

Performance Metric	Training	Test
True Positive Rate	22%	55%
False positive rate	4%	19%
Model accuracy	79%	79%

Table 5.4: *Performance Metrics for Random Forrest*

5.b Gradient Boosting (GB)

The second model is a gradient boosting model with a Bernoulli loss function (meant for 0 or 1 type classification problems) based on 300 trees or iterations (because from figure 5.1 above it's

clear that after 300 trees there is only marginal improvements in error reduction). In this model 39 of the 63 predictor variables (62%) had non-zero influence. These variables are shown below in figure 5.3 in their relative order of influence. Due to space constraints not all the 39 variables are shown below. The top 3 variables are:

- PAY_RT2
- PAY_AMT1
- TOTAL_PAY_RT

When compared with the random forest model (TOTAL_PAY_RT, PAY_AMT1, PAY_AMT2), it looks like 2 out of the 3 top 3 variables are the same.

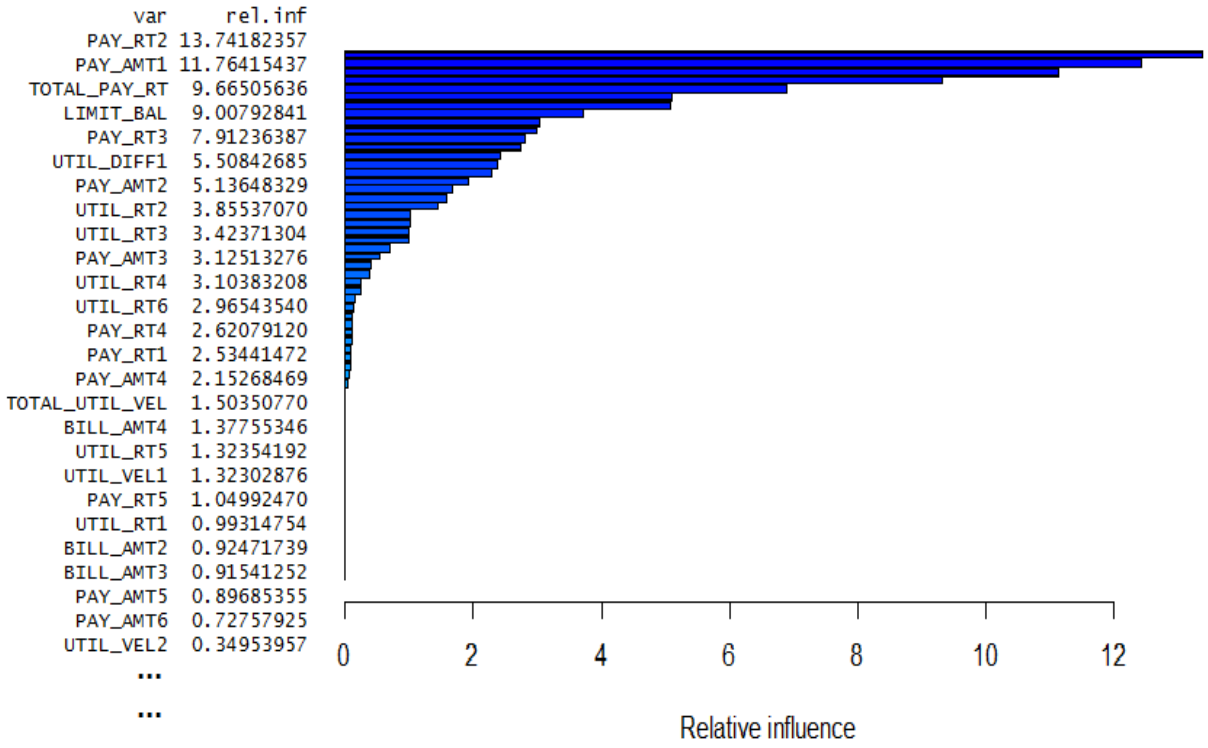

Figure 5.3: *Variable Importance Plot – Gradient Boosting (reformatted manually)*

Here is the performance summary for the gradient boosting model:

TRAIN	Actual Non-Default	Actual Default
Predicted Non-Default	11728	3312
Predicted Default	29	111

TEST	Actual Non-Default	Actual Default
Predicted Non-Default	5744	1513
Predicted Default	22	44

Table 5.5: *Confusion Matrix for Gradient Boosting (training and test)*

Performance Metric	Training	Test
True Positive Rate	79%	67%
False positive rate	22%	21%
Model accuracy	78%	79%

Table 5.6: *Performance Metrics for Gradient Boosting*

5.c Logistic Regression with Backwards Variable Selection

The previous Random Forest and Gradient Boosting models identified a pool of interesting predictor variables:

Gradient Boosting	Random Forrest
PAY_RT2	TOTAL_PAY_RT
PAY_AMT1	PAY_AMT1
TOTAL_PAY_RT	PAY_AMT2
LIMIT_BAL	PAY_RT2
PAY_RT3	AGE
UTIL_DIFF1	UTIL_VEL1
PAY_AMT2	PAY_RT1
PAY_AMT3	TOTAL_UTIL_VEL
UTIL_RT2	UTIL_RT1
UTIL_RT4	LIMIT_BAL

Table 5.7: *Top 10 variables from Random Forrest and Gradient Boosting models*

We can use that information to help choose an initial pool of predictor variables for the next model which is a logistic regression model, which includes: TOTAL_PAY_RT, PAY_RT1, PAY_RT2, PAY_RT3, PAY_AMT1, PAY_AMT2, PAY_AMT3, LIMIT_BAL, AGE, UTIL_DIFF1, UTIL_RT1, UTIL_RT2, UTIL_RT4, UTIL_VEL1 and TOTAL_UTIL_VEL. We then employ the backwards variable selection technique amongst these variables (which is a "variable selection procedure in which all variables are entered into the equation and then sequentially removed. The variable with the smallest partial correlation with the dependent variable is considered first for removal. If it meets the criterion for elimination, it is removed. After the first variable is removed, the variable remaining in the equation with the smallest partial correlation is considered next. The procedure stops when there are no variables in the equation that satisfy the removal criteria" *Ref: IBM Knowledge Center [1]*) to arrive at a more optimal logistic regression model. Following is a table of the model coefficients and their p-values.

Variables	Coefficient	Standard Error	p-value
Intercept	-1.42E+00	1.04E-01	< 2e-16
TOTAL_PAY_RT	7.45E-01	1.73E-01	1.67E-05
PAY_RT2	-2.25E-01	9.61E-02	0.019232
PAY_RT3	-3.33E-01	9.50E-02	0.000465
PAY_AMT1	-2.59E-05	3.80E-06	9.06E-12
PAY_AMT2	-1.48E-05	3.36E-06	9.88E-06
PAY_AMT3	-4.20E-06	2.37E-06	0.075695
LIMIT_BAL	-1.97E-06	2.16E-07	< 2e-16
AGE	1.03E-02	2.11E-03	9.64E-07
UTIL_DIFF1	-3.14E-01	2.02E-01	0.119898
UTIL_RT1	-4.14E-01	2.42E-01	0.08733
UTIL_RT2	7.82E-01	2.33E-01	0.000787
UTIL_RT4	4.67E-01	1.52E-01	0.002143
TOTAL_UTIL_VEL	-1.80E+00	8.57E-01	0.035427

Table 5.8: *Table of Logistic Regression coefficients and p-values*

Here is the performance summary for the logistic regression model:

TRAINING	Actual Non-Default	Actual Default
Predicted Non-Default	11757	3423
Predicted Default	0	0

TEST	Actual Non-Default	Actual Default
Predicted Non-Default	11728	3312
Predicted Default	29	111

Table 5.9: *Confusion Matrix for Logistic Regression (training and test)*

Performance Metric	Training	Test
True Positive Rate	0%	79%
False positive rate	23%	22%
Model accuracy	77%	78%

Table 5.10: *Performance Metrics for Logistic Regression*

5.d Support Vector Machine (SVM)

The last model is a support vector machine model with a non-linear radial kernel and optimized for binary classification with a cost of 1 and gamma of 0.01492537. The cost parameter determines the soft margin (that allows some examples to be ignored for a better overall model fit – in this case nothing is ignored) and gamma determines the bias-variance trade-off (precision vs. accuracy trade-off). Generally speaking large gammas leaf to high bias and low variance models. Since there are a lot of variables it is difficult to show a margin plot that separates the correctly classified records from the incorrectly classified ones on a 2 dimension x-axis vs. y-axis graph. Instead the following performance plot shows how the hyperplane was optimized by

looking at different combinations of cost and epsilon, which indicate a margin of tolerance for errors or penalties. In this plot, the darker the region, the best the model:

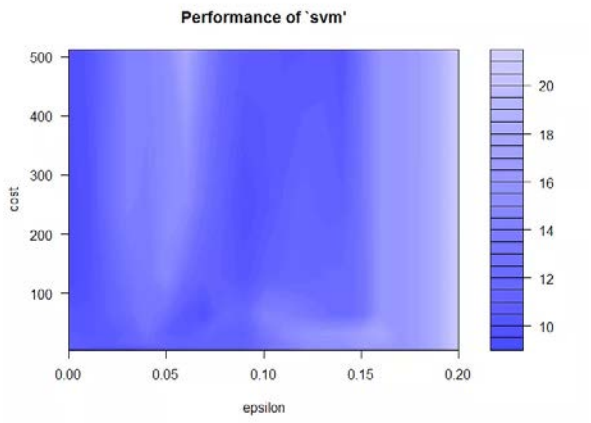

Figure 5.4: *Performance Plot - SVM*

Here is the performance summary for the SVM model:

TRAINING	Actual Non-Default	Actual Default
Predicted Non-Default	9261	2314
Predicted Default	2656	751

TEST	Actual Non-Default	Actual Default
Predicted Non-Default	558	77
Predicted Default	259	77

Table 5.9: *Confusion Matrix for SVM (training and test)*

Performance Metric	Training	Test
True Positive Rate	22%	23%
False positive rate	20%	12%
Model accuracy	67%	65%

Table 5.10: *Performance Metrics for SVM*

6. Comparison of Results

Following are the aggregated results from section 5 for all the models:

	Train			Test		
Model	Model Accuracy	False Positive	True Positive Rate	Model Accuracy	False Positive	True Positive Rate
Random Forrest	79%	4%	22%	79%	19%	55%
Gradient Boosting	78%	22%	79%	79%	21%	67%
Logistic Regression	77%	23%	0%	78%	22%	79%
Support Vector Machine	67%	20%	22%	65%	12%	23%

Table 6.1: *Performance metrics for all models*

From the plots in figures 6.1 and 6.2 below which are based on the table above, we see that the model accuracy is highest in both the training and test data for the first random forest model and it declined as we experimented with different modelling approaches. However model accuracy is not the end and be all when it comes to measuring performance and we also have to consider true sensitivity (true positive rate) and false negative rate. When these 2 additional metrics are considered we notice the following:

	Training data	Test data
Model with best (highest) true positive rate	Gradient Boosting	Logistic Regression
Model with best (lowest) false positive rate	Random Forest	Support Vector Machine

Table 6.2: *Nuanced interpretation of the performance metrics*

In fact in the training data set, the Gradient Boosting methods seems to have the best overall balance of all three worlds: there is a marginal degradation in overall model accuracy but it comes with a much better model sensitivity (true positive) rate than the Random Forest model (in fact it has the best model sensitivity amongst all 4 models) and a slightly worse false positive

rate around 20% which is not that different among the remaining 3 models (Gradient Boosting, Logistic Regression and Support Vector Machine).

Carrying over this analogy in the test data set, the plain and simple logistic regression seems to have the best overall balance of all three worlds: there is a marginal degradation in overall model accuracy but it comes with a much better model sensitivity (true positive) rate than the Random Forest model (in fact it has the best model sensitivity amongst all 4 models) and a slightly worse false positive rate which is not that different for 3 of the models (Random Forest, Gradient Boosting and Logistic Regression).

In other words, Gradient Boosting seems to have the best overall performance in the training data but logistic regression has the best overall performance in the test data set, which may then be the best performing and simplest model to deploy in a production environment.

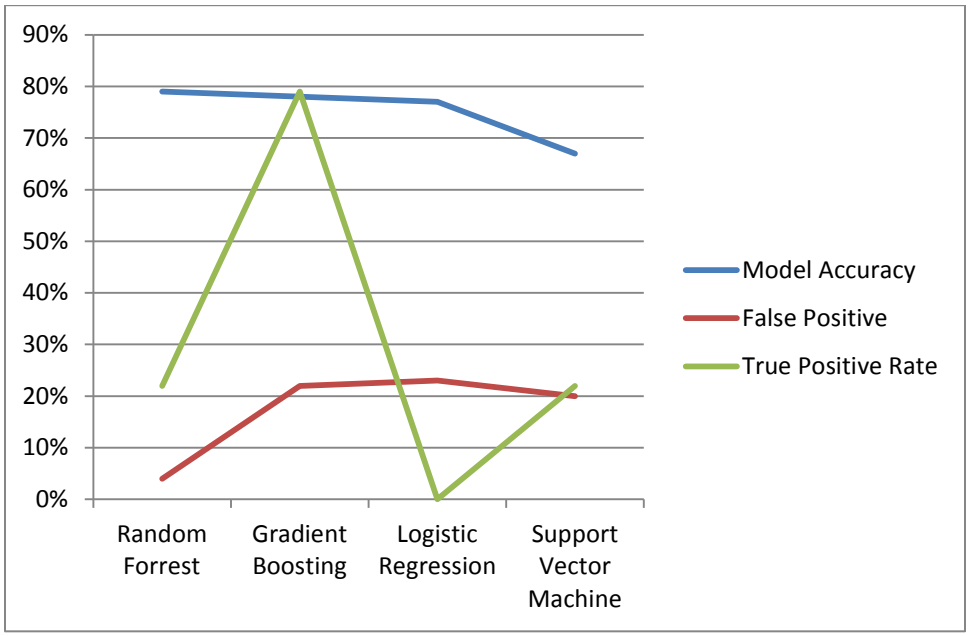

Figure 6.1: *Performance – Training Data*

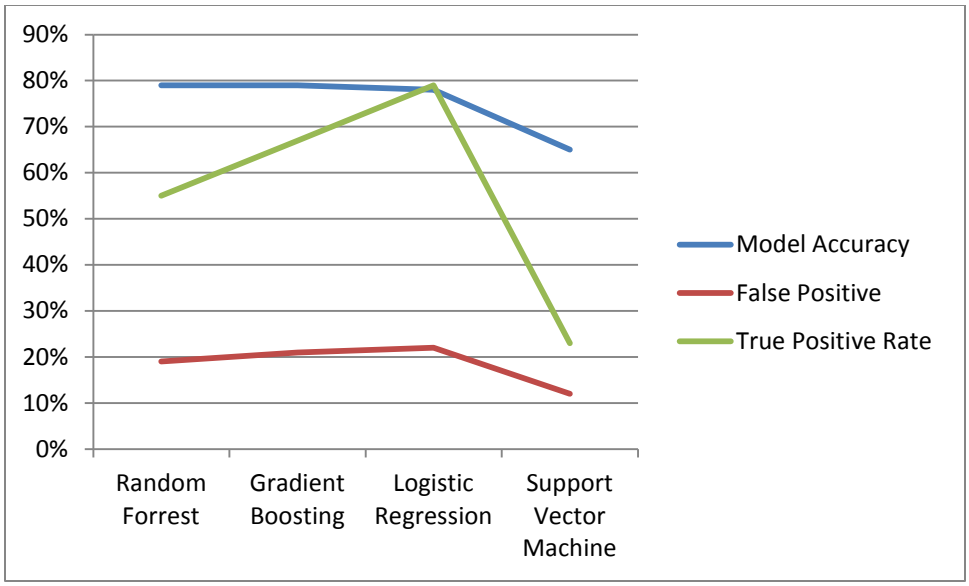

Figure 6.2: *Performance – Test Data*

7. Conclusions

In this capstone project we attempted to predict credit card defaults using a data set that contained the credit card payment history of 30,000 Taiwanese customers of a financial institution for 6 consecutive months in 2005. The dataset also contained various socio-demographic attributes like age, marital status, education level and others. As part of this exercise we first segmented the data into a training, test and validation set and then applied some consistent data quality controls and engineered new features that allowed for benchmarking of across individual credit card payment histories using normalized rates. We then applied four different common predictive modeling techniques on the data which were random forest, gradient boosting, logistic regression with variable selection and support vector machine.

Of these four models, random forest yielded the highest model accuracy (79% in both training and test data sets) but the simplest model, which was the logistic regression model, had a

better combination of model sensitivity (79%) and false positive rate (22%) with marginal degradation of model accuracy (78%) in the test data set. This result seems to align with the common belief among statisticians that the simplest models are sometimes the best performing and the easiest to deploy in a production environment.

Although the scope of this capstone project was limited to 4 different predictive models, if further improvements in model performance were to be attempted in the future, then it might be well worth the effort to experiment with neural network based modelling techniques. It is also quite reasonable to expect that combining the data set with other data points; if possible, such as income and bank balance might lead to an increase in model performance. Additionally further performance tuning based on different cost-optimization functions within the existing four models and experimenting with additional engineered features where the data cleansing rules are slightly more nuanced (for example instead of simply flagging out of range values as missing, simply replacing these values to the mode or median value to avoid skewing the data unintentionally) might also lead to improvements in overall model performance.

8. Bibliography

UCI Machine Learning Repository (2016, January 26). Default of credit card client's data set. Retrieved February 26, 2018, from https://archive.ics.uci.edu/ml/datasets/default%20of%20credit%20card%20clients

Investopedia. (n.d.). Credit Card Utilization Ratio. Retrieved February 26, 2018, from https://www.investopedia.com/terms/c/credit-utilization-rate.asp

Rickert, J. (2013, June 19). Draw nicer Classification and Regression Trees with the rpart.plot package. Retrieved February 26, 2018, from http://blog.revolutionanalytics.com/2013/06/plotting-classification-and-regression-trees-with-plotrpart.html

Charpentier, A. (2015, June 17). 'Variable Importance Plot' and Variable Selection. Retrieved February 26, 2018, from https://www.r-bloggers.com/variable-importance-plot-and-variable-selection/

Wikipedia. Out-of-bag error. Retrieved February 26, 2018, from https://en.wikipedia.org/wiki/Out-of-bag_error

Markham, K. (2014, March 25). Simple guide to confusion matrix terminology. Retrieved February 26, 2018, from http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

Colakoglu, C. et. al. (2013). Kaggle: Implications of Out-of-bag (OOB) error in Random Forest models (Titanic: Machine Learning from Disaster). Retrieved February 26, 2018, from https://www.kaggle.com/c/titanic/discussion/3554

IBM Knowledge Center. (n.d.). Linear Regression Variable Selection Methods. Retrieved February 26, 2018, from https://www.ibm.com/support/knowledgecenter/en/SSLVMB_23.0.0/spss/base/linear_regression_methods.html

KDnuggets. (n.d.). An Introduction to Random Forests for Beginners. Retrieved February 26, 2018, from https://www.kdnuggets.com/2014/03/introduction-random-forests-free-ebook.html

Helal, M. (2016, November 7). Quora: How do you explain decision tree and random forests, in layman's terms? Retrieved February 26, 2018, from https://www.quora.com/How-do-you-explain-decision-tree-and-random-forests-in-layman%E2%80%99s-terms

Pistre, G. (2016, October 19). Quora: How does XGBoost (gradient Boosting) compare with Random Forest? Retrieved February 26, 2018, from https://www.quora.com/How-does-XGBoost-gradient-Boosting-compare-with-Random-Forest

Galkin, A. et. al. (2011, November 28). Stack Exchange: Bagging, boosting and stacking in machine learning. Retrieved February 26, 2018, from https://stats.stackexchange.com/questions/18891/bagging-boosting-and-stacking-in-machine-learning

Banmrick, N. (2016, June 24). AYLIEN: Support Vector Machines for dummies; A Simple Explanation. Retrieved February 26, 2018, from http://blog.aylien.com/support-vector-machines-for-dummies-a-simple/

Girard, J. M. (2016, April 22). Quora: What are C and gamma with regards to a support vector machine? Retrieved February 26, 2018, from https://www.quora.com/What-are-C-and-gamma-with-regards-to-a-support-vector-machine

9. Appendices

9.1 Summary of the original data set in R

```
################################################################################
# 'data.frame':   30000 obs. of  30 variables:
#  $ ID       : int  1 2 3 4 5 6 7 8 9 10 ...
#  $ LIMIT_BAL : int  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
#  $ SEX      : int  2 2 2 2 1 1 1 2 2 1 ...
#  $ EDUCATION : int  2 2 2 2 2 1 1 2 3 3 ...
#  $ MARRIAGE  : int  1 2 2 1 1 2 2 2 1 2 ...
#  $ AGE      : int  24 26 34 37 57 37 29 23 28 35 ...
#  $ PAY_0    : int  2 -1 0 0 -1 0 0 0 0 -2 ...
#  $ PAY_2    : int  2 2 0 0 0 0 0 -1 0 -2 ...
#  $ PAY_3    : int  -1 0 0 0 -1 0 0 -1 2 -2 ...
#  $ PAY_4    : int  -1 0 0 0 0 0 0 0 0 -2 ...
#  $ PAY_5    : int  -2 0 0 0 0 0 0 0 0 -1 ...
#  $ PAY_6    : int  -2 2 0 0 0 0 0 -1 0 -1 ...
#  $ BILL_AMT1 : int  3913 2682 29239 46990 8617 64400 367965 11876 11285 0 ...
#  $ BILL_AMT2 : int  3102 1725 14027 48233 5670 57069 412023 380 14096 0 ...
#  $ BILL_AMT3 : int  689 2682 13559 49291 35835 57608 445007 601 12108 0 ...
#  $ BILL_AMT4 : int  0 3272 14331 28314 20940 19394 542653 221 12211 0 ...
#  $ BILL_AMT5 : int  0 3455 14948 28959 19146 19619 483003 -159 11793 13007 ...
#  $ BILL_AMT6 : int  0 3261 15549 29547 19131 20024 473944 567 3719 13912 ...
#  $ PAY_AMT1  : int  0 0 1518 2000 2000 2500 55000 380 3329 0 ...
#  $ PAY_AMT2  : int  689 1000 1500 2019 36681 1815 40000 601 0 0 ...
#  $ PAY_AMT3  : int  0 1000 1000 1200 10000 657 38000 0 432 0 ...
#  $ PAY_AMT4  : int  0 1000 1000 1100 9000 1000 20239 581 1000 13007 ...
#  $ PAY_AMT5  : int  0 0 1000 1069 689 1000 13750 1687 1000 1122 ...
#  $ PAY_AMT6  : int  0 2000 5000 1000 679 800 13770 1542 1000 0 ...
#  $ DEFAULT   : int  1 1 0 0 0 0 0 0 0 0 ...
#  $ u        : num  0.288 0.788 0.409 0.883 0.94 ...
#  $ train    : num  1 0 1 0 0 1 0 0 0 1 ...
#  $ test     : num  0 0 0 0 0 0 1 0 1 0 ...
#  $ validate  : num  0 1 0 1 1 0 0 1 0 0 ...
#  $ data.group: num  1 3 1 3 3 1 2 3 2 1 ...
# ################################################################################
# Note: data.group values are constructed to partition the data set in a single dimension;
# data.group <- 1*my.data$train + 2*my.data$test + 3*my.data$validate;
# Use the train, test, and validate flags to define the train, test, and validate data sets;
################################################################################
```

9.2 Finalized data preparation setps in R

```
rm(list=ls())

# Define path and file;
# You will need to change the path value to match the location on your own computer;
my.path <- 'C:\\Users\\amustaki\\Documents\\mspa\\498\\data\\';
my.file <- paste(my.path,'credit_card_default.RData',sep='');
```

```
# Read the RData object using readRDS();
credit_card_default <- readRDS(my.file)
#Backup original data set
###############
credit_card_default_bkp <- credit_card_default

#New variables
###############

#Mute EDUCAION for unknown values
credit_card_default$EDUCATION[credit_card_default$EDUCATION==0] <- NA
credit_card_default$EDUCATION[credit_card_default$EDUCATION==5] <- NA
credit_card_default$EDUCATION[credit_card_default$EDUCATION==6] <- NA

#Mute MARRIAGE for unknown values
credit_card_default$MARRIAGE[credit_card_default$MARRIAGE==0] <- NA

#Mute PAY_X for unknown values
credit_card_default$PAY_0[credit_card_default$PAY_0==-2] <- NA
credit_card_default$PAY_0[credit_card_default$PAY_0==0] <- NA

credit_card_default$PAY_2[credit_card_default$PAY_2==-2] <- NA
credit_card_default$PAY_2[credit_card_default$PAY_2==0] <- NA

credit_card_default$PAY_3[credit_card_default$PAY_3==-2] <- NA
credit_card_default$PAY_3[credit_card_default$PAY_3==0] <- NA

credit_card_default$PAY_4[credit_card_default$PAY_4==-2] <- NA
credit_card_default$PAY_4[credit_card_default$PAY_4==0] <- NA

credit_card_default$PAY_5[credit_card_default$PAY_5==-2] <- NA
credit_card_default$PAY_5[credit_card_default$PAY_5==0] <- NA

credit_card_default$PAY_6[credit_card_default$PAY_6==-2] <- NA
credit_card_default$PAY_6[credit_card_default$PAY_6==0] <- NA

#Engineered Features - Utilization Rate
util_rate_func <- function (x,y){
 ifelse( y != 0 ,
       ifelse(x>y,1,x/y),
       ifelse(x>0,1,0)
 )
}

credit_card_default$UTIL_RT1 <- util_rate_func(credit_card_default$BILL_AMT1, credit_card_default$LIMIT_BAL)
credit_card_default$UTIL_RT2 <- util_rate_func(credit_card_default$BILL_AMT2, credit_card_default$LIMIT_BAL)
credit_card_default$UTIL_RT3 <- util_rate_func(credit_card_default$BILL_AMT3, credit_card_default$LIMIT_BAL)
credit_card_default$UTIL_RT4 <- util_rate_func(credit_card_default$BILL_AMT4, credit_card_default$LIMIT_BAL)
credit_card_default$UTIL_RT5 <- util_rate_func(credit_card_default$BILL_AMT5, credit_card_default$LIMIT_BAL)
credit_card_default$UTIL_RT6 <- util_rate_func(credit_card_default$BILL_AMT6, credit_card_default$LIMIT_BAL)

#PAyment Rate
pay_rate_func <- function (x,y) {
```

```
 ifelse( x!=0,
       ifelse(y>x,1,y/x),
       1
 )
}

credit_card_default$PAY_RT1 <- pay_rate_func(credit_card_default$BILL_AMT1,credit_card_default$PAY_AMT1)
credit_card_default$PAY_RT2 <- pay_rate_func(credit_card_default$BILL_AMT2,credit_card_default$PAY_AMT2)
credit_card_default$PAY_RT3 <- pay_rate_func(credit_card_default$BILL_AMT3,credit_card_default$PAY_AMT3)
credit_card_default$PAY_RT4 <- pay_rate_func(credit_card_default$BILL_AMT4,credit_card_default$PAY_AMT4)
credit_card_default$PAY_RT5 <- pay_rate_func(credit_card_default$BILL_AMT5,credit_card_default$PAY_AMT5)
credit_card_default$PAY_RT6 <- pay_rate_func(credit_card_default$BILL_AMT6,credit_card_default$PAY_AMT6)


#Utilization Velocity
credit_card_default$UTIL_VEL1 = credit_card_default$UTIL_RT1 - credit_card_default$UTIL_RT2
credit_card_default$UTIL_VEL2 = credit_card_default$UTIL_RT2 - credit_card_default$UTIL_RT3
credit_card_default$UTIL_VEL3 = credit_card_default$UTIL_RT3 - credit_card_default$UTIL_RT4
credit_card_default$UTIL_VEL4 = credit_card_default$UTIL_RT4 - credit_card_default$UTIL_RT5
credit_card_default$UTIL_VEL5 = credit_card_default$UTIL_RT5 - credit_card_default$UTIL_RT6


#Total Utilization Velocity
credit_card_default$TOTAL_UTIL_VEL = ((credit_card_default$UTIL_VEL1 +
                          credit_card_default$UTIL_VEL2 +
                          credit_card_default$UTIL_VEL3 +
                          credit_card_default$UTIL_VEL4 +
                          credit_card_default$UTIL_VEL5)/5)



#Difference from min utilization
credit_card_default <- transform(credit_card_default, min_UTIL_RT = pmin(UTIL_RT1,
                                          UTIL_RT2,
                                          UTIL_RT3,
                                          UTIL_RT4,
                                          UTIL_RT5,
                                          UTIL_RT6
))

credit_card_default$UTIL_DIFF1 = credit_card_default$UTIL_RT1 - credit_card_default$min_UTIL_RT
credit_card_default$UTIL_DIFF2 = credit_card_default$UTIL_RT2 - credit_card_default$min_UTIL_RT
credit_card_default$UTIL_DIFF3 = credit_card_default$UTIL_RT3 - credit_card_default$min_UTIL_RT
credit_card_default$UTIL_DIFF4 = credit_card_default$UTIL_RT4 - credit_card_default$min_UTIL_RT
credit_card_default$UTIL_DIFF5 = credit_card_default$UTIL_RT5 - credit_card_default$min_UTIL_RT
credit_card_default$UTIL_DIFF6 = credit_card_default$UTIL_RT6 - credit_card_default$min_UTIL_RT



#Total PAy Rate
credit_card_default$TOTAL_PAY_RT = ((credit_card_default$PAY_RT1 +
                         credit_card_default$PAY_RT2 +
                         credit_card_default$PAY_RT3 +
```

```
                        credit_card_default$PAY_RT4 +
                        credit_card_default$PAY_RT5 +
                        credit_card_default$PAY_RT6)/6)

#Age bins
credit_card_default$AGE_60_ABOVE <- as.numeric(credit_card_default$AGE>=60)
credit_card_default$AGE_45_59 <- as.numeric( credit_card_default$AGE>= 45 & credit_card_default$AGE<60)
credit_card_default$AGE_40_44 <- as.numeric( credit_card_default$AGE>= 40 & credit_card_default$AGE<45)
credit_card_default$AGE_36_39 <- as.numeric( credit_card_default$AGE>= 36 & credit_card_default$AGE<40)
credit_card_default$AGE_33_35 <- as.numeric( credit_card_default$AGE>= 33 & credit_card_default$AGE<36)
credit_card_default$AGE_30_32 <- as.numeric( credit_card_default$AGE>= 30 & credit_card_default$AGE<33)
credit_card_default$AGE_29_BELOW <- as.numeric(credit_card_default$AGE < 30)

credit_card_default$AGE_BIN <- with(credit_card_default, ifelse(
  credit_card_default$AGE_60_ABOVE > 0, '60_and_above', ifelse(
    credit_card_default$AGE_45_59 > 0, '45_to_59', ifelse(
      credit_card_default$AGE_40_44 > 0, '40_to_44', ifelse(
        credit_card_default$AGE_36_39 > 0, '36_to_39', ifelse(
          credit_card_default$AGE_33_35 > 0, '33_to_35', ifelse(
            credit_card_default$AGE_30_32 > 0, '30_to_32', '29_and_below'
          )
        )
      )
    )
  )
 ) ) )



#Convert discrete variables to factor allow EDA plotting
credit_card_default$SEX <- as.factor(credit_card_default$SEX)
credit_card_default$EDUCATION <- as.factor(credit_card_default$EDUCATION)
credit_card_default$MARRIAGE <- as.factor(credit_card_default$MARRIAGE)
credit_card_default$PAY_0 <- as.factor(credit_card_default$PAY_0)
credit_card_default$PAY_2 <- as.factor(credit_card_default$PAY_2)
credit_card_default$PAY_3 <- as.factor(credit_card_default$PAY_3)
credit_card_default$PAY_4 <- as.factor(credit_card_default$PAY_4)
credit_card_default$PAY_5 <- as.factor(credit_card_default$PAY_5)
credit_card_default$PAY_6 <- as.factor(credit_card_default$PAY_6)

#Class boundaries
######################
credit_card_default$UTIL_RT2_0.49_AND_ABOVE <- as.numeric(credit_card_default$UTIL_RT2 >= 0.49)
credit_card_default$UTIL_RT2_BELOW_0.49 <- as.numeric(credit_card_default$UTIL_RT2 < 0.49)

credit_card_default$UTIL_RT3_0.45_AND_ABOVE <- as.numeric(credit_card_default$UTIL_RT3 >= 0.45)
credit_card_default$UTIL_RT3_BELOW_0.45 <- as.numeric(credit_card_default$UTIL_RT3 < 0.45)

credit_card_default$UTIL_RT4_0.48_AND_ABOVE <- as.numeric(credit_card_default$UTIL_RT4 >= 0.48)
credit_card_default$UTIL_RT4_BELOW_0.48 <- as.numeric(credit_card_default$UTIL_RT4 < 0.48)

credit_card_default$UTIL_RT5_0.68_AND_ABOVE <- as.numeric(credit_card_default$UTIL_RT5 >= 0.68)
credit_card_default$UTIL_RT5_BELOW_0.68 <- as.numeric(credit_card_default$UTIL_RT5 < 0.68)
```

```
credit_card_default$UTIL_RT6_0.62_AND_ABOVE <- as.numeric(credit_card_default$UTIL_RT6 >= 0.62)
credit_card_default$UTIL_RT6_BELOW_0.62 <- as.numeric(credit_card_default$UTIL_RT6 < 0.62)


credit_card_default$TOTAL_PAY_RT_0.18_AND_ABOVE <- as.numeric(credit_card_default$TOTAL_PAY_RT >=
0.18)
credit_card_default$TOTAL_PAY_RT_BELOW_0.18 <- as.numeric(credit_card_default$TOTAL_PAY_RT < 0.18)

credit_card_default$LIMIT_BAL_125k_AND_ABOVE <- as.numeric(credit_card_default$LIMIT_BAL >= 125000)
credit_card_default$LIMIT_BAL_BELOW_125k <- as.numeric(credit_card_default$LIMIT_BAL < 125000)

credit_card_default$UTIL_RT2_BIN <- with(credit_card_default, ifelse(
  credit_card_default$UTIL_RT2_0.49_AND_ABOVE > 0
  , '0.49_AND_ABOVE'
  , 'BELOW_0.49'
)
)

credit_card_default$UTIL_RT3_BIN <- with(credit_card_default, ifelse(
  credit_card_default$UTIL_RT3_0.45_AND_ABOVE > 0
  , '0.45_AND_ABOVE'
  , 'BELOW_0.45'
)
)

credit_card_default$UTIL_RT4_BIN <- with(credit_card_default, ifelse(
  credit_card_default$UTIL_RT4_0.48_AND_ABOVE > 0
  , '0.48_AND_ABOVE'
  , 'BELOW_0.48'
)
)

credit_card_default$UTIL_RT5_BIN <- with(credit_card_default, ifelse(
  credit_card_default$UTIL_RT5_0.68_AND_ABOVE > 0
  , '0.68_AND_ABOVE'
  , 'BELOW_0.68'
)
)

credit_card_default$UTIL_RT6_BIN <- with(credit_card_default, ifelse(
  credit_card_default$UTIL_RT6_0.62_AND_ABOVE > 0
  , '0.62_AND_ABOVE'
  , 'BELOW_0.62'
)
)


credit_card_default$TOTAL_PAY_RT_BIN <- with(credit_card_default, ifelse(
  credit_card_default$TOTAL_PAY_RT_0.18_AND_ABOVE > 0
  , '0.18_AND_ABOVE'
  , 'BELOW_0.18'
)
)
```

```
credit_card_default$LIMIT_BAL_125k_AND_ABOVE <- as.numeric(credit_card_default$LIMIT_BAL >= 125000)
credit_card_default$LIMIT_BAL_BELOW_125k <- as.numeric(credit_card_default$LIMIT_BAL < 125000)

credit_card_default$LIMIT_BAL_BIN <- with(credit_card_default, ifelse(
  credit_card_default$LIMIT_BAL_125k_AND_ABOVE > 0
  , '125k_AND_ABOVE'
  , 'BELOW_125k'
)
)

cc_train <- credit_card_default[credit_card_default$train == 1,]

#remove temporary and ineffective variables
drop <- c("ID",
        "PAY_0",
        "PAY_2",
        "PAY_3",
        "PAY_4",
        "PAY_5",
        "PAY_6",
        "u",
        "train",
        "test",
        "validate",
        "data.group",
        "min_UTIL_RT",
        "AGE_BIN",
        "UTIL_RT2_BIN",
        "UTIL_RT3_BIN",
        "UTIL_RT4_BIN",
        "UTIL_RT5_BIN",
        "UTIL_RT6_BIN",
        "TOTAL_PAY_RT_BIN",
        "LIMIT_BAL_BIN"
)

cc_train = cc_train[,!(names(cc_train) %in% drop)]

cc_test <- credit_card_default[credit_card_default$test == 1,]
```