EC4308 Machine Learning And Economic Forecasting

SEMESTER I 2020-2021

Applying Machine Learning Techniques to Predict Credit Default Risk

Andy Low Wei Liang [redacted]

01 Introduction:

Credit Risk Assessment is the practice of assessing and minimising the probability of a borrower's inability to repay a loan obligation. This has proven itself to be a long time challenge for financial institutions due to several reasons such as lack of data and tedious reporting processes. Over the last decade, many financial institutions have since developed models in an attempt to minimise loss arising from default payments by refining their loan approval decision processes. Hence, this project aims to create a model to predict the probability of a client defaulting in payment based on several characteristics such as previous payment history of clients and demographics of the client. This project is especially useful for creditors as they can better profile prospective customers based on their default risk and issue higher credit limits or loans to customers with lower risk. Creditors can also track the default risk of existing customers to aid them in considering further extension of loans.

Many machine learning methods have been applied to credit risk prediction. Data sets for credit risks are usually noisy or data sets are small, since financial institutions generally do not share data to form large data sets. Resultantly, it is difficult to train models, since a useful database with a great amount of data cannot be formed (Twala, 2010). However, in Twala, 2010, we see that ensemble (combining) classifiers have the potential to increase prediction accuracy. In our paper, we will examine ensemble classifiers and other standard classification methods to propose a model that gives accurate prediction.

In this paper, we aim to develop a model capable of identifying clients who are likely to default in credit payment based on several variables. The 'Default of Credit Card Clients' dataset (Yeh, 2016) contains 30,000 observations and 25 variables

02 Exploratory Data Analysis (EDA):

2.1 Data Overview

The dataset used contains information on default payments, demographic factors, credit data, history of payment and bill statements of credit card clients in Taiwan from April 2005 to September 2005. We aim to predict the probability of a client defaulting in payment based on several characteristics such as previous payment history of clients and demographics of the client. The data consists of 24 independent variables and 1 dependent variable, default.payment.next.month (Detailed explanation of each variable is in the Appendix) . The dependent variables are :

ID,LIMIT_BAL,SEX,EDUCATION,MARRIAGE,AGE,PAY_0,PAY_1,PAY_2,PAY_3,PAY_4,PAY_5,PAY_6,PAY_AMT1,PAY_AMT2,PAY_AMT3,PAY_AMT4,PAY_AMT5,PAY_AMT6,BILL_AMT1,BILL_AMT2,BILL_AMT3,BILL_AMT4,BILL_AMT5,BILL_AMT6

The check for null values showed that none exists.

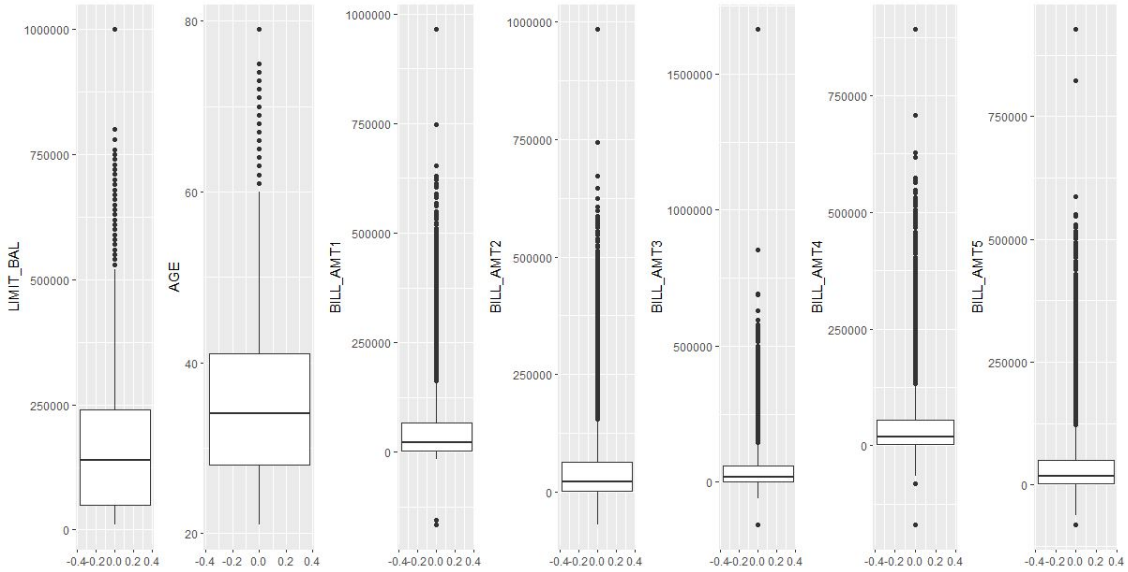
```
> sum(is.na.data.frame(data))
[1] 0
> sum(is.null(data))
[1] 0
```

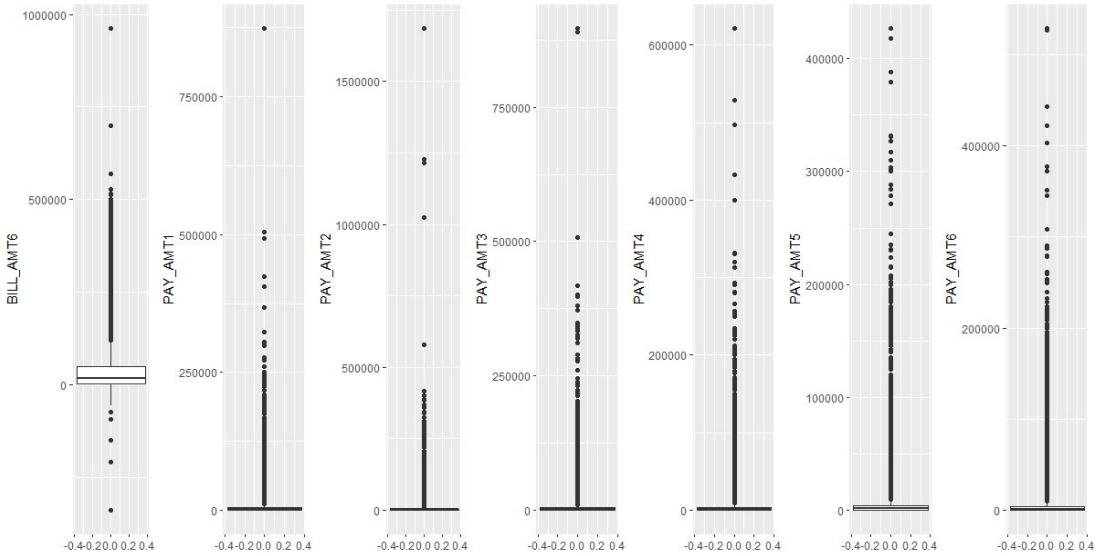
2.2 Inconsistent Values within the Dataset

```
EDUCATION MARRIAGE  PAY_0      PAY_2      PAY_3      PAY_4      PAY_5      PAY_6
0:   14   0:   54   -2: 2759   -2: 3782   -2: 4085   -2: 4348   -2: 4546   -2: 4895
1:10585   1:13659   -1: 5686   -1: 6050   -1: 5938   -1: 5687   -1: 5539   -1: 5740
2:14030   2:15964   0 :14737   0 :15730   0 :15764   0 :16455   0 :16947   0 :16286
3: 4917   3:  323   1 : 3688   1 :   28   1 :    4   1 :    2   2 : 2626   2 : 2766
4:  123              2 : 2667   2 : 3927   2 : 3819   2 : 3159   3 :  178   3 :  184
5:  280              3 :  322   3 :  326   3 :  240   3 :  180   4 :   84   4 :   49
6:   51              4 :   76   4 :   99   4 :   76   4 :   69   5 :   17   5 :   13
                     5 :   26   5 :   25   5 :   21   5 :   35   6 :    4   6 :   19
                     6 :   11   6 :   12   6 :   23   6 :    5   7 :   58   7 :   46
                     7 :    9   7 :   20   7 :   27   7 :   58   8 :    1   8 :    2
                     8 :   19   8 :    1   8 :    3   8 :    2
```

From the range of unique values above, we observed some inconsistent values such as -2 in PAY_n, 3 in marriage and 0,5 and 6 in education. We subsequently resolved these values with clarification from the author.

2.3 Distribution of Continuous Variables

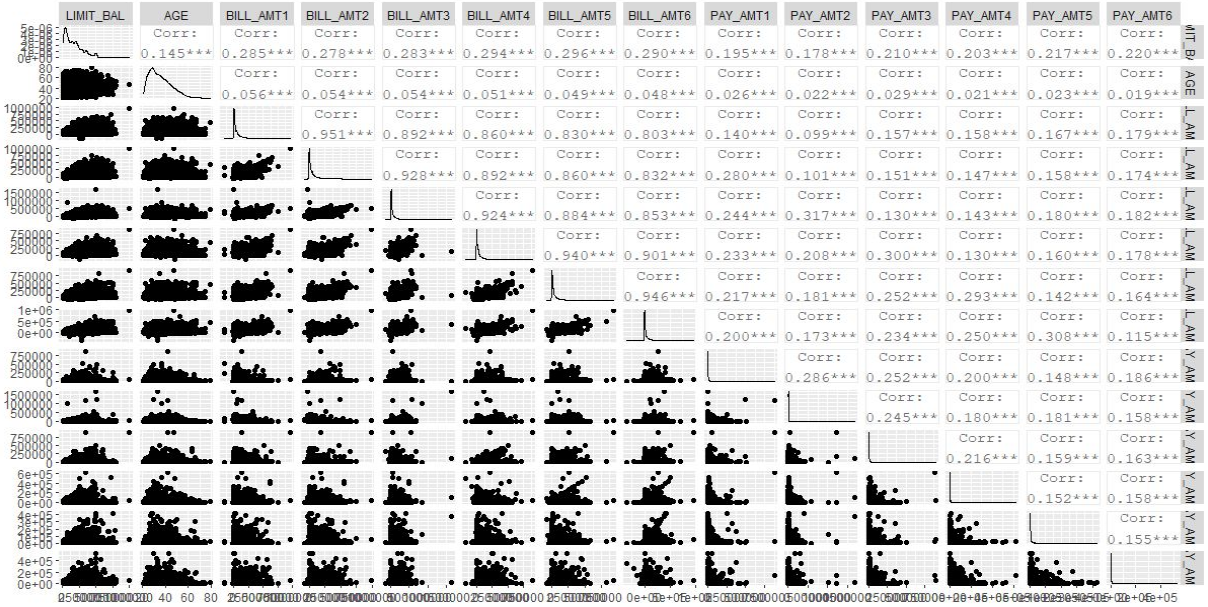




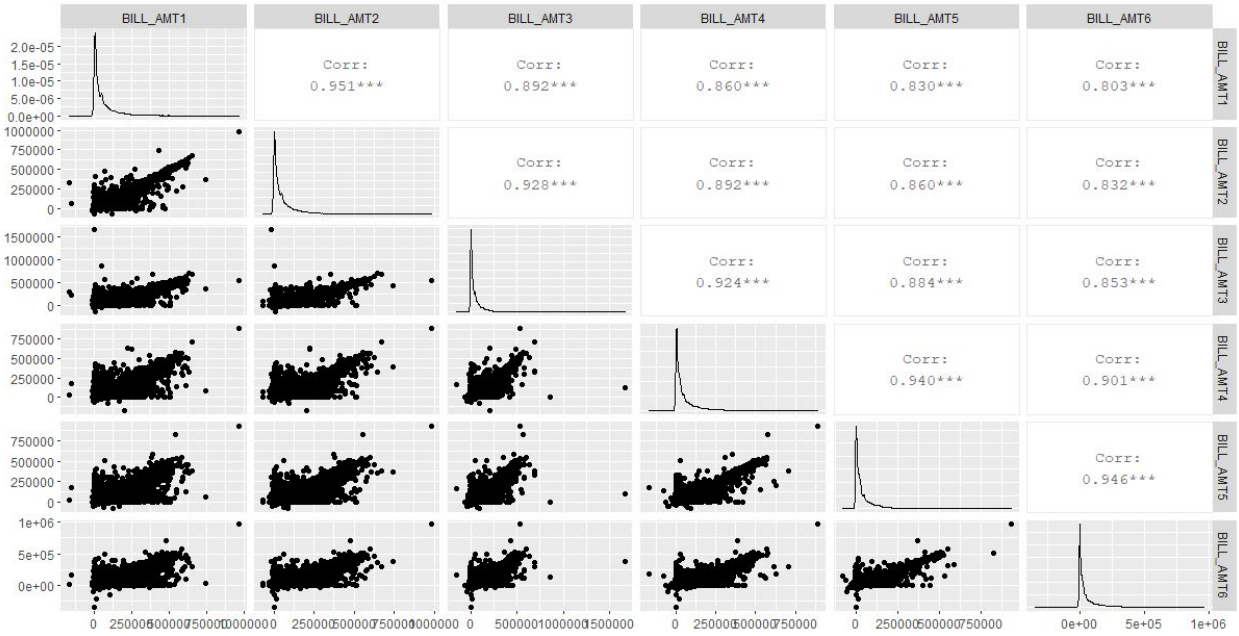
Continuous variables are also plotted in box plots shown below, allowing us to identify outliers and trends between months. Interestingly, the 25th to 75th percentile range of PAY_n columns are close to zero, much smaller compared to other columns. This signifies the need for data normalisation.

In addition, outliers are consistently observed across all columns and we have decided that it was not an entry error, as the selected outlier was perhaps granted higher credit limits hence higher debts and payments.

2.4 Correlation of Variables

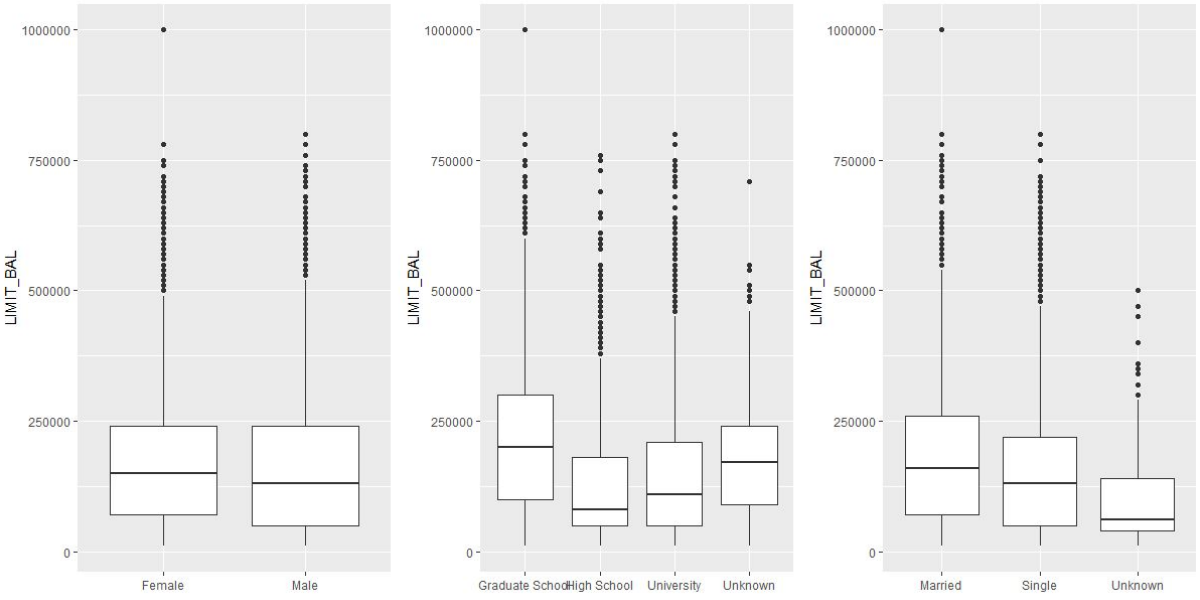


The overall correlation between variables is plotted to observed multicollinearity. Particularly high levels of correlation between BILL columns were observed.

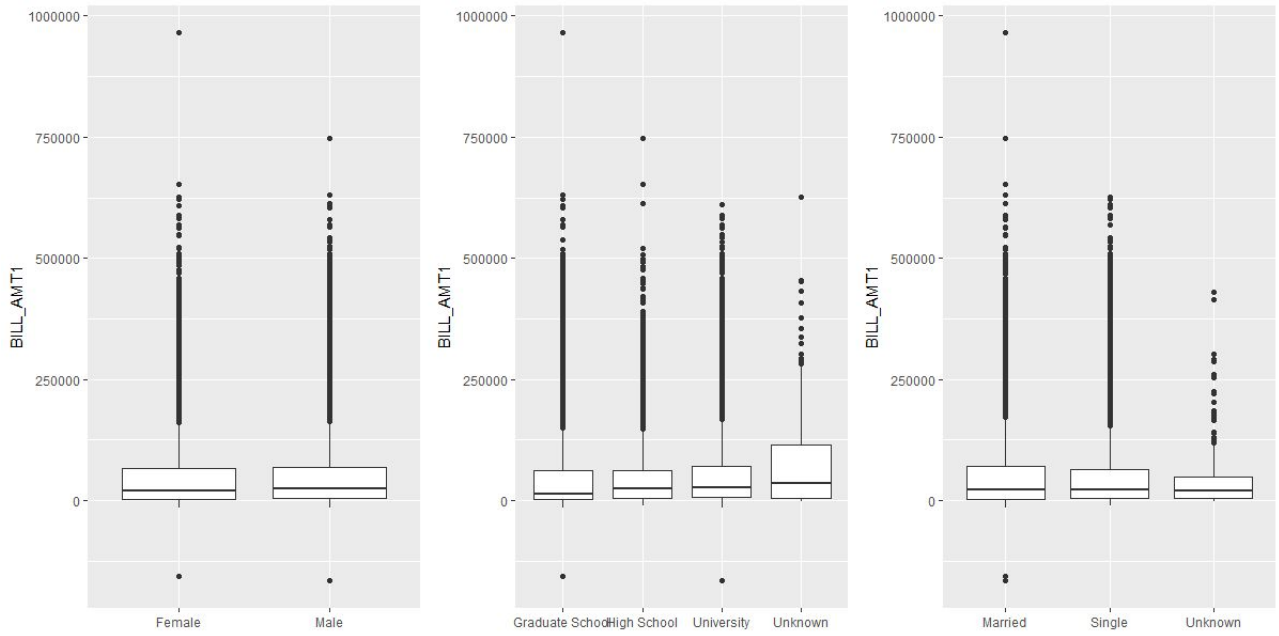


Hence, we zoomed into the correlation plot for bill amount and observed that correlation falls as the month observed is further. It makes logical sense for bill amounts to have a higher correlation with months closer to it as compared to months further.

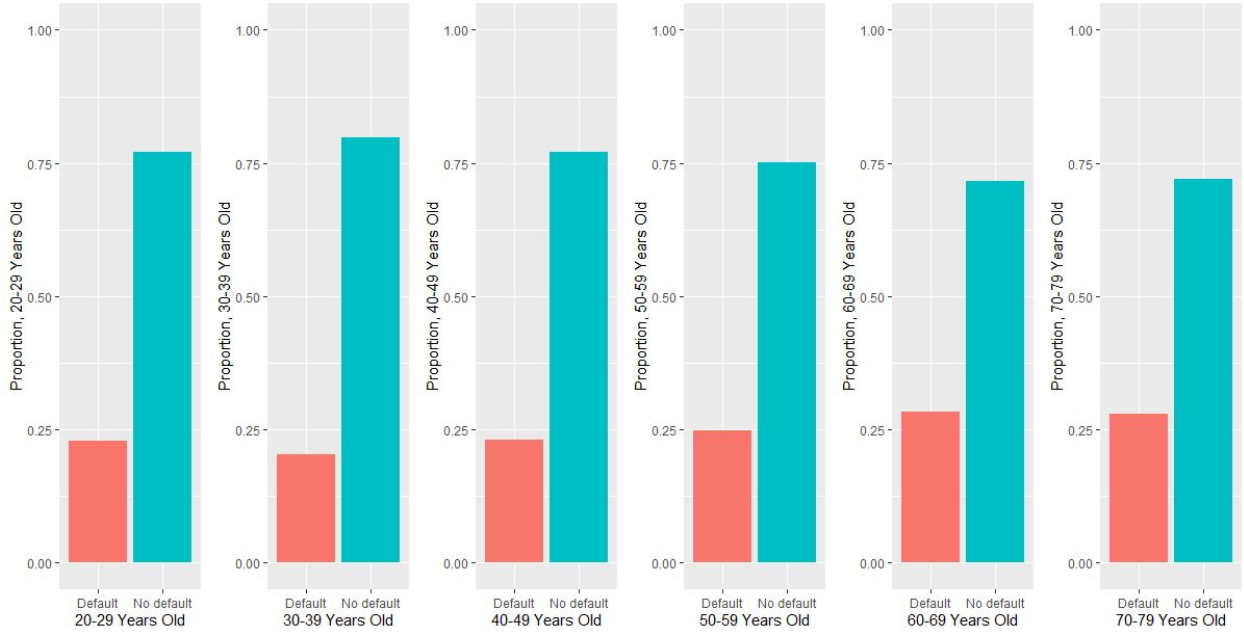
2.5 Observing Correlation in Distributions



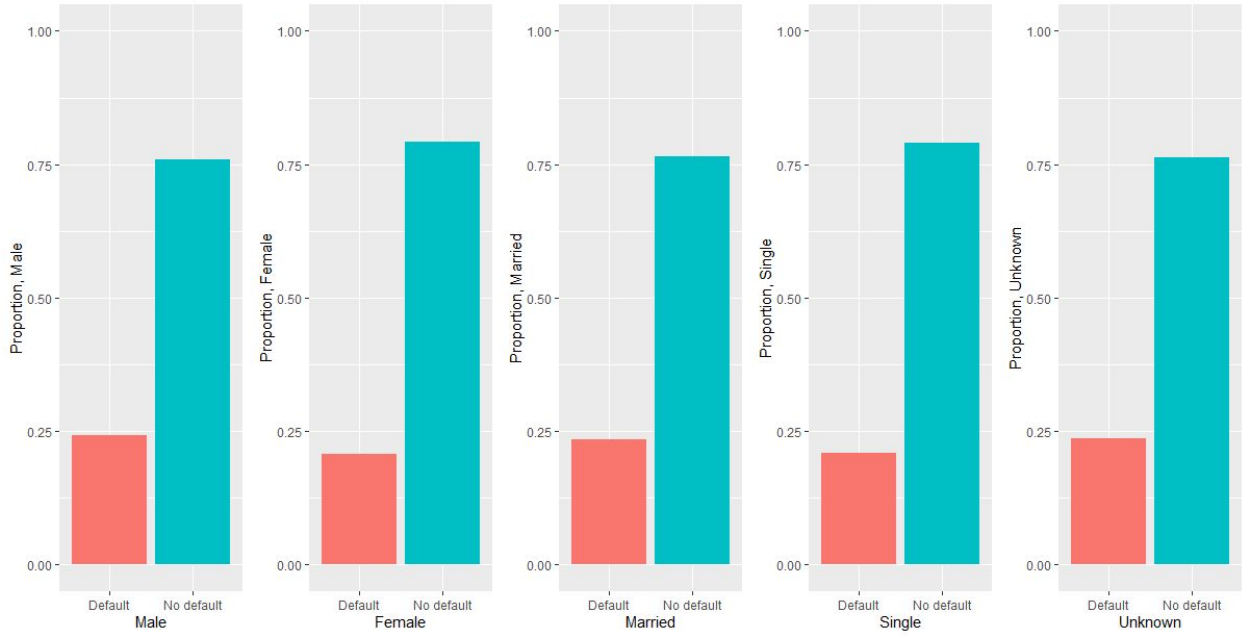
Correlation of plot of limit with demographics, education and marital status seems to show some sort of correlation.



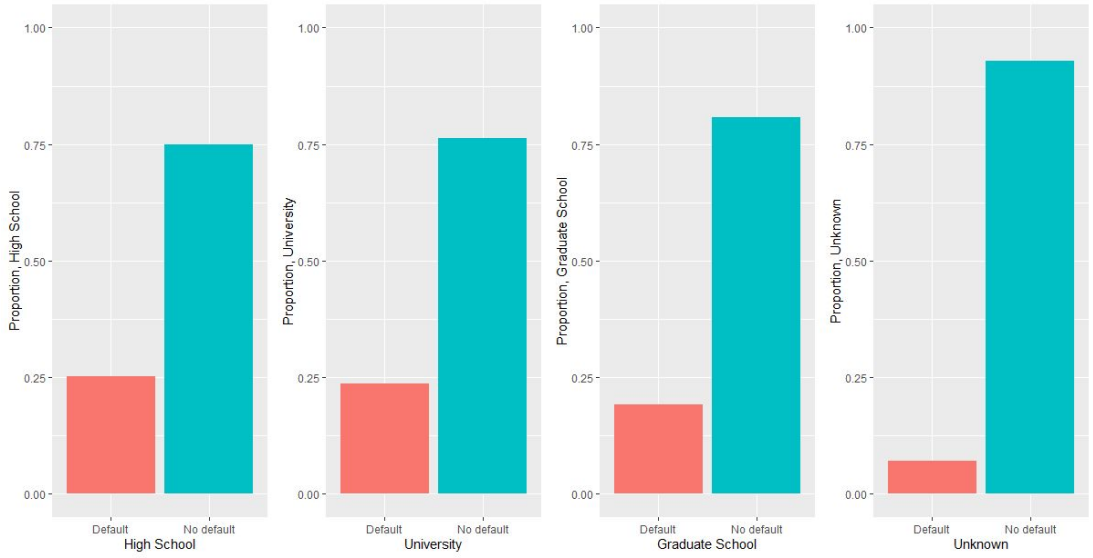
Similar box plots are also plotted for BILL_AMT1:BILL_AMT6 and PAY_AMT1:PAY_AMT6. Box plot indicates that the 12 variables are not highly correlated with the demographics variable. One example from BILL_AMT1 is shown.



Histogram of proportion of defaulters for different age groups. Some correlation as the default proportion seems to increase with age group.



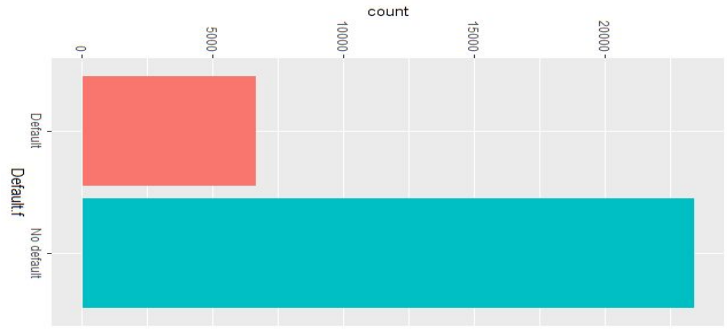
The proportion of default/no default for different gender and marital status does not show much correlation between gender and marriage status with defaulting.



Proportion of default/no default for different education groups. Lower default proportion for individuals who are more highly educated.

2.6 Dataset Evaluation

2.6.1 Skewed classes



Data set is imbalanced with more data points for no default (23364) as compared to default at only 22.1% (6636). This may lower prediction accuracy for defaulters as the algorithm has less data points to train on for defaulters.

03 Data Pre-processing:

3.1 Change of PAY_0 to PAY_1

Changing column PAY_0 to PAY_1 as a more intuitive form of column naming.

3.2 Removal of ID column

Removing column 'ID' as it has no correlation with the dependent variable in the study.

3.3 Ignoring multicollinearity in variables

There are high levels of collinearity between the variables concerning Bill_Amt and low to moderate levels of collinearity exist for all other variables. This can contribute to a high variance model which can worsen the quality of our forecast. However, in feature selection, we found that most of the bill variables are dropped and thus, multicollinearity would not be as big an issue.

3.4 Feature Scaling (Data Normalisation)

Feature scaling standardizes the range of values of features. From the boxplots above, the large range of the features were observed. Hence feature scaling will help to ensure that features of varying ranges will not be favoured by our models and is crucial to all machine learning methods, as it heavily impacts the weights and coefficients in the later stages. This will help our prediction models be more unbiased.

04 Feature Selection and Models:

4.1 Logistic Regression

Logistic Regression is a classification model where the dependent variable can only take two values, 0 (false) and 1 (true) (Twala, 2010), perfect for our dependent variable of default or not. However, it is important to note that Logistic Regression will not provide exact values of 0 or 1. Instead, it returns the probabilistic value which lies between 0 and 1 (Gurucharan, 2020).

Under the Logistic Regression, instead of a regression line, we fit a S-shaped curve with the formula:

$$1/(1 + e^{-value})$$

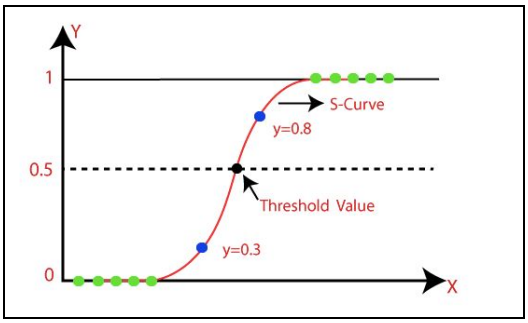


Figure 1: Logistic Function (Source: Gurucharan, 2020)

As seen in Figure 1, the logistic function can take on any values approaching 0 or 1 but never exactly. In Linear Regression, when predicting if the client will default their payments, we would get continuous values between 0 and 1. However, under the Logistic Regression model, we set a threshold of 0.5 as seen in Figure 1. As such, the single client will be classified as "default" if the value is greater than the threshold of 0.5, and "will not default" if the value is less than 0.5

Least Absolute Shrinkage and Selection Operator (LASSO)

LASSO is a method that selects features, and regularizes model parameters by shrinking coefficients, with a penalty that is the sum of the absolute value of the coefficients, sometimes shrinking them exactly to 0 when lambda is large enough, whereas Ridge Regression does not. The feature selection occurs after shrinkage, where only non-zero coefficients are selected. Under subset selection, we enjoy an interpretable model, but suffer from high variance. Whereas under Ridge Regression, we obtain predictions with low variance, but at the cost of reduced interpretability. LASSO hence aims to retain the advantages of subset selection and ridge regression, while mitigating the drawbacks such that LASSO results in models with better prediction accuracy while granting greater interpretability.

We will be using features selected by LASSO and use it in the logistic regression model. The features selected are:

LIMIT_BAL,SEX,EDUCATION,MARRIAGE,AGE,PAY_1,PAY_2,PAY_3,PAY_4,PAY_5,PAY_6,BILL_AMT1,PAY_AMT1,PAY_AMT2,PAY_AMT4,PAY_AMT5,PAY_AMT6.

Recursive Partitioning and Regression Trees (Rpart)

The Rpart algorithm works by splitting the dataset recursively. The subsets that arise from the splits are further split until a predetermined termination criterion is fulfilled. At each step, the split is made based on the independent variable that results in the largest possible reduction in heterogeneity of the dependent variables.

We will be using features selected by the Rpart algorithm. The features selected are PAY_1, PAY_2, PAY_3, PAY_4, PAY_5 out of the 23 predictors.

Feature selection	Test MSE	Accuracy	ROC
Without any feature selection	**0.144**	**0.823**	**0.713**
LASSO	**0.144**	**0.822**	**0.712**
Rpart	**0.145**	**0.826**	**0.699**

4.2 Principal Component Regression & Partial Least Squares Regression

4.2.1 Principal Component Regression (PCR)

Principal Components Regression (PCR) is a technique that aims to tackle the issue of multicollinearity of a dataset. Instead of regressing Y on X, PCR performs linear regressions on the principal components derived from Principal Component Analysis (PCA). To understand PCR, we need to first understand how PCA works.

PCA works with a large set of correlated variables to learn about the predictors, without any relation to Y. PCA aims to transform features that are correlated into principal components. Principal components are linear combinations of the original features, and each principal component is uncorrelated to each other (He, 2020). In this way, 2 or more features are bundled together such that the number of features used is decreased. This allows us to explain most of the variance in the data using fewer number of principal components than the number of features we have.

Geometrically, looking at Figure 2, the first principal component explains the greatest amount of variance and is hence known as the largest principal component. The first principal component is closest to the

data points. The second principal component, orthogonal to the first, explains the next greatest amount of variance. The first principal component is uncorrelated with the second principal component.

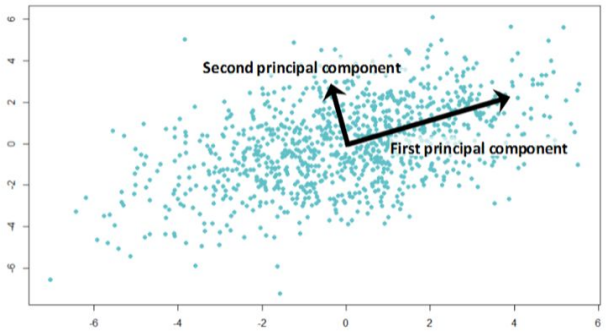


Figure 2: Principal Components in PCA (Source: Analytics Vidhya)

The downside of using PCR is that certain useful signals may be hidden in the lower principal components, and may end up being discarded. To solve this issue, we can run both PCR and LASSO on the independent variables X in the dataset to find if there were features not picked up by PCR.

Method	Test MSE	Accuracy	ROC
One-sigma	0.152	0.812	0.696
Randomization	**0.150**	**0.819**	**0.705**
Lasso (λ = 1SE)	0.151	0.817	0.700

4.2.2 Partial Least Squares Regression (PLSR)

Partial Least Squares Regression (PLSR) is an alternative to PCR, where PLSR tries to account for both the dependent variable Y and the independent variable X.

Under PLSR, the model tries to find a multidimensional direction that can explain variance of X, as well as the correlation with Y. To find the first partial least squares direction denoted as Z_1, PLSR regresses Y on each individual X while also assigning weights denoted as Φ_{1j}. Each predictor X is then regressed on Z_1 to derive the remaining information (known as residuals) not explained by Z_1. We can then find the second partial least squares direction denoted as Z_2 by regressing the dependent variable Y on the residuals (Hastie et al., 2009). We then continue this step until we reach M, which is the smaller value of the number of predictors, or sample size.

Referring to Figure 3, ignoring that the Figure 3 is based on the context of advertising data, we can see that PCR (reflected by dotted line) explains more on the variance of X and is also more focused on X.

Whereas PLS, (reflected by the solid line) shows us a different direction as compared to the PCR line. PCR tries to explain X best, whereas PLS tries to also account for Y.

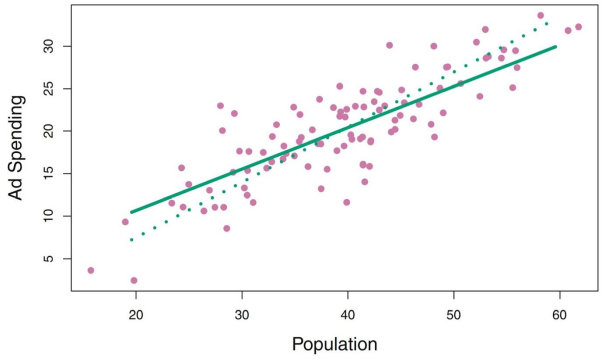


Figure 3: Partial Least Squares Regression (PLSR) vs Principal Component Regression (PCR). (Source: James et al. (2014), ISLR, Springer)

Running PLSR can help us improve predictability by ensuring that selected principal components are now also correlated with our dependent variable Y.

Method	**Test MSE**	**Accuracy**	**ROC**
1SE	0.150	0.815	0.705
Randomization	**0.150**	**0.815**	**0.707**

4.3. Recursive Partitioning and Regression Trees & Random Forest

4.3.1 Recursive Partitioning and Regression Trees (Rpart)

Rpart is an algorithm that allows us to grow classification and regression trees. Under rpart, we only consider binary partitions. To do this, we first split our data into two regions as seen in the left panel in Figure 4, where the first split is denoted by t_1. We then continue to split the regions, deciding where the spilt will take place by taking the spilt that minimizes the squared loss. We refer to these regions (R_1, R_2,...,R_5) as terminal nodes or leaves of the tree. We repeatedly split the regions, until we reach a stopping criterion, such as when the minimum number of nodes are reached.

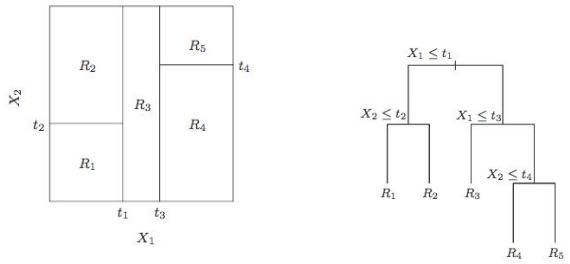


Figure 4 : Binary Partitions and Decision Tree under Rpart (Source: Hastie et al. (2009): Elements of Statistical Learning, Springer)

To better visualise how our regions in the left panel corresponds to a tree, we refer to the right panel in the same Figure 4. Our previous partitions can then be depicted by a single tree, where at each node, we go left if $X_1 \leq t_1$, and right if $X_1 \geq t_1$. We follow this process and go down the tree until we reach a leaf. The advantage of using a single tree is the ease of interpretability. This benefit however, disappears as we go on to explore Random Forest.

Method	Test MSE	Accuracy	ROC
Anova	**0.135**	**0.826**	**0.748**

4.3.2 Bagged Trees and Random Forest

Random Forest is an ensemble method that builds on the bagging technique, providing an improvement to averaging by considering trees that are uncorrelated. We first draw bootstrap samples (random sample from the dataset with replacement), then fit a tree for each bootstrap sample. As decision trees vary with each change in training set chosen, it may return us different trees with different splits every time we try to generate a new tree. As such, Random Forest takes advantage of this and settles the final classification decision by taking the majority vote. As seen in Figure 5, Random Forest unlike in Rpart, generates multiple decision trees where each decision tree selects, at random, less than the number of predictors available while bagged trees includes all predictors. This results in lower correlation across trees, and hence lowering the variance while maintaining the low bias of any particular tree.

Random Forest improves our predictability, but at the cost of reducing interpretability as compared to Rpart as we are now faced with more than one decision tree.

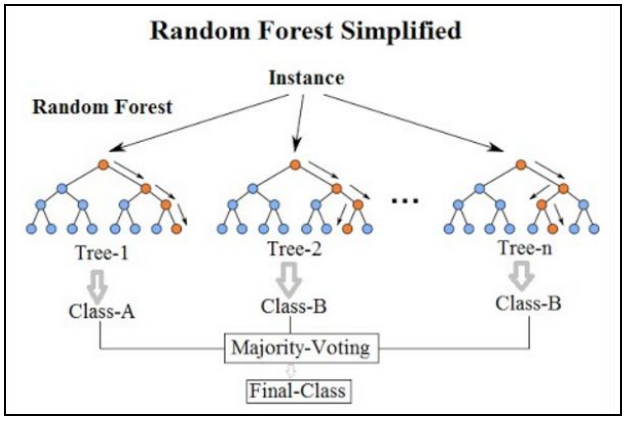


Figure 5 : Random Forest (Source: Jagannath, 2020)

No of trees	Max nodes	mtry	Test MSE	Accuracy	ROC
5000	29	23	0.133	0.826	0.748
5000	**29**	**7**	**0.133**	**0.826**	**0.769**

4.3.3 Gradient Boosting Machine (GBM)

Gradient Boosting is a forward stagewise method that gradually improves a weak learner, producing a prediction model in the form of weaker prediction models. Gradient Boosting consists of three elements, a loss function to be optimised, a weak learner to predict and an additive model to add weak learners while minimising the loss function. Firstly, we consider a tree that minimises our desired loss function. Additively, we fit a tree to a model that reduces the loss function, partitioning the tree, where existing trees remain unaltered. We achieve this by parameterising the tree, modifying the parameters then moving towards reducing the residual loss.

The algorithm is as follows:

1. Initialize $f_0(x) = argmin_y \sum_{i=1}^{N} L(y_i, \gamma)$
2. For m = 1 to M
   a. Compute $r_{im} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_1)}]_{f=f_{m-1}}$ for i = 1, … , N
   b. Fit regression tree to $\{r_{im}\}$ giving terminal partitions $\{R_{jm}\}$, for $j = 1, \ldots, J_m$
   c. For $j = 1, \ldots, J_m$, compute $\gamma_{jm} = argmin_\gamma \Sigma_{x_i \varepsilon R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$
   d. Update model to $f_m(x) = f_{m-1}(x) + \lambda \sum_{j=1}^{J_m} \gamma_{jm} 1(x \varepsilon R_{jm})$

Method	Test MSE	Accuracy	ROC
CV	0.132	0.826	0.775
OOB	**0.133**	**0.827**	**0.771**

4.5 Neural Network

Neural Networks (NN) are nonlinear regression models capable of making mathematical decisions, as well as possess the ability to handle both regression and classification tasks. NN can be represented by network diagrams as shown in Figure 6, where a neural network consists of 3 types of layers: input, hidden and output layer. Each input node is an independent variable that is multiplied by a weight that has been chosen to make the model fit the training data well. The input nodes are then transmitted in a single direction through the hidden layer, which is not directly observed. The input nodes are then subsequently transformed into output notes, otherwise also known as target or dependent variables.

A neural network can have more than 1 hidden layer. More generally, a neural network that has only 1 hidden layer is called Shallow Neural Network. Whereas a neural network with more than 1 hidden layer is known as Deep Neural Network (Simpson, 2018). As an example, our network diagram in Figure 6 is an example of a Deep Neural Network with 3 hidden layers. A Deep Neural Network also helps to increase the predictive power.

In reality, many of our inputs and outputs have relationships that are non-linear and complex. Neural Network is a powerful technique that is able to model such relationships, and still return us accurate predictions (Mahanta, 2017). Although we do have to be careful when using the NN model because it is usually overparameterized, and hence easily overfits.

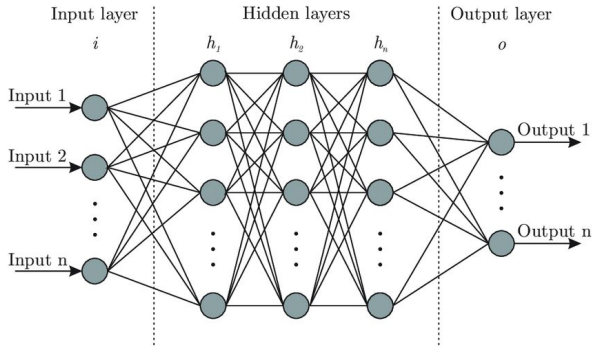


Figure 6: Network Diagram (Source: Shukla, 2020).

Method	Test MSE	Accuracy	ROC
Without Feature Selection	0.178	0.822	0.762
Selecting important features	**0.177**	**0.823**	**0.761**

05 Conclusion:

After evaluating the different models with various feature selection and reduction methods, we have concluded that the best model is Gradient Boosting Machine (GBM). However, in the tradeoff between a higher accuracy of 82.7% against a lower test MSE of 0.132 and higher ROC of 0.776, GBM with the number of iterations chosen by CV is decidedly the best.

Interestingly, logistic regression and decision trees in general work exceptionally well in this dataset, predicting defaulters with a comparably high accuracy and ROC. A reason for such could be due to the nature of our classification problem being of only two classes.

Model	**Test MSE**	**Accuracy**	**ROC**
Logistic Regression	0.144	0.823	0.713
Logistic Regression(LASSO)	0.144	0.822	0.712
Logistic Regression(Rpart)	0.145	0.826	0.699
PCR (1 sigma)	0.152	0.812	0.696
PCR (min CV)	0.150	0.819	0.705
PCR w/ LASSO (λ = 1SE)	0.151	0.817	0.700
PLS (1SE)	0.150	0.815	0.705
PLS (min CV)	0.150	0.815	0.707
Decision Tree (Anova)	0.135	0.826	0.748
Bagged Trees (ntree = 5000, mtry = 23)	0.133	0.826	0.748
Random Forest (ntree = 5000, mtry = 7)	0.133	0.826	0.769
GBM (CV)	**0.132**	**0.826**	**0.776**
GBM (OOB)	**0.133**	**0.827**	**0.771**
Neural Network (w/o Feature Selection)	0.178	0.822	0.762
Neural Network (w/ Feature Selection)	0.177	0.823	0.761

06 Room for Improvement:

6.1 Data Collection

The data set is rather limited in size, with only 30000 observations. This can impede the predictive ability of the model. In addition, the models require 6 months worth of data in order to generate forecasts resulting in a long and cumbersome data collection process. Data also has to be fed on a rolling basis into the model as each month passes by which requires some form of automation as well.

The current dataset is also heavily limited in features beyond credit debt status such as BILL, PAY, LIMIT_BAL. Other variables which characterises the individual may be useful in predicting defaulters. One of such could be the income of the individual. An individual with higher income has a greater capacity to repay his credit card debt and thus, is less likely to default.

Another useful variable can be the credit utilisation percentage, measuring the percentage of available credit the individual is using or credit rating. An individual with a high credit utilisation percentage or bad credit rating would more likely default since he is taking on more debt proportionally.

6.2 Presence of Outliers

In order to ensure that the dataset conforms to reality, several outliers are included in our training models. Referencing the box plots in the EDA section, both the bill amount and pay amount variables have a very long right tail and there are some visible outliers from the boxplot. Such outliers are expected as selected individuals may be granted higher credit limits and hence higher debts and payment consequentially.

6.3 Class Imbalance

Understandably the data set is imbalanced with more data points for no default (23364) as compared to defaulters at only 22.1% (6636) as consumers tend to avoid the high interest charged by credit companies. As such, the model tends to be biased towards non-defaulters as there are insufficient observations of defaulters for model training. In return, the models produced are adept in picking up non-defaulters. From the confusion matrix below, we can see that our trained models will only be accurate when predicting default.payment.next.month = 0. Hence, our models will be more accurate if the distribution of the dependent variable is more balanced.

	pcr.pred_1	
yTest	0	1
0	4709	1
1	1286	4

To overcome the class imbalance, the package ubSMOTE was utilised to carry out the synthetic minority over-sampling technique (SMOTE) on the minority class of defaulters. SMOTE utilises the K Nearest Neighbours concept, to generate new instances from existing minority cases supplied as input. Hence, it aims to increase the number of observations in the dataset in a balanced way. It has also been shown to increase test accuracy (Cinaroglu, 2020). However, it worked conversely in our case, leading to overly generalised models unable to predict both non-defaulters and defaulters reliably, as shown in the confusion matrix below.

yTest	pcr.pred_1 0	pcr.pred_1 1
0	4445	265
1	813	477

Another issue is that accuracy, defined as the total number of correct predictions over the total number of predictions, can no longer be used as a performance evaluation metric. A model that predicts all the observations as no default can still have a relatively high level of accuracy. Thus, we focused on other metrics like precision and recall through ROC instead.

6.4 Hybrid Methods and Ensemble Learning Across Models

We can further improve our model by assuming uncorrelated forecasts of the various models and attaching empirical weights (Bates & Granger, 1969)

In addition, we can also make use of the regression method introduced by Grander and Ramanathan (1984) or LASSO to combine forecasts with varying weights

Unfortunately in this case, our best models are decision trees and running hybrid methods on such requires large computational power and time.

6.5 Parameter Tuning

In practice, neural networks show great promise in producing results of higher accuracy and unbiasedness. However, time and computational power is required to properly tune the parameters to an optimised state. The numerous ongoing research on convolutional neural networks and deep neural networks is a testament to this.

6.6 Feature Engineering

By observing and comparing our approach to those on kaggle. Feature engineering is possible for this dataset but provides little if any benefits as seen in Singh (2018) attempt at grouping AGE, BAL_LIMIT and summing PAY, with an end accuracy of only 0.804 using Neural Networks while ours achieved 0.8225 on average. However, we still believe that more creative features can possibly be engineered provided sufficient unique features.

07 References

Analytics Vidhya. (2016, March 21). PCA: A Practical Guide to Principal Component Analysis in R & Python. Retrieved from https://www.analyticsvidhya.com/blog/2016/03/pca-practical-guide-principal-component-analysis-python/

Bates, J. M., & Granger, C. W. (1969). The Combination of Forecasts. *Journal of the Operational Research Society, 20*(4), 451-468. doi:10.1057/jors.1969.103

Cinaroglu, S. (2020). The impact of oversampling with "ubSMOTE" on the performance of machine learning classifiers in prediction of catastrophic health expenditures. *Operations Research for Health Care, 27*, 100275. doi:10.1016/j.orhc.2020.100275

Granger, C. W., & Ramanathan, R. (1984). Improved methods of combining forecasts. Journal of Forecasting, 3(2), 197-204. doi:10.1002/for.3980030207

Gurucharan, M. (2020, August 06). Machine Learning Basics: Logistic Regression. Retrieved November 10, 2020, from https://towardsdatascience.com/machine-learning-basics-Logistic-regression-890ef5e3a272

Hastie, T., Tibshirani, R., Friedman, J. (2009). The Elements of Statistical Learning, Springer.

He, K. (2020, February 23). SVD in Machine Learning: PCA. Retrieved from https://towardsdatascience.com/svd-in-machine-learning-pca-f25cf9b837ae

Jagannath, V. (2020, August 06). Random Forest Template for TIBCO Spotfire. Retrieved from https://community.tibco.com/wiki/random-forest-template-tibco-spotfire

James, G., Witten, D., Hastie, T., Tibshirani, R. (2017). An introduction to statistical learning with applications in R. New York: Springer.

James, G., Witten, D., Hastie, T., Tibshirani, R. (2014). An introduction to statistical learning with applications in R. New York: Springer.

Mahanta, J. (2017, July 10). Introduction to Neural Networks, Advantages and Applications. Retrieved from https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-Applications-96851bd1a207

Shukla, L. (2020). Weights and Biases. Retrieved from https://www.kdnuggets.com/2019/11/designing-neural-networks.html

Simpson, M. (2018, November 19). Machine Learning Algorithms: What is a Neural Network? Retrieved from https://www.verypossible.com/insights/machine-learning-algorithms-what-is-a-neural-network

Singh, D. (2018, August 02). CC defaulters - NNET&Parameter Tuning Tutorial. Retrieved November 10, 2020, from https://www.kaggle.com/digvijaysingh16/cc-defaulters-nnet-parameter-tuning-tutorial/notebook

Twala, B. (2010). Multiple classifier application to credit risk assessment. *Expert Systems with Applications, 37*(4), 3326-3336. doi:10.1016/j.eswa.2009.10.018

Yeh, I. (2016, January 26). Default of credit card clients Data Set. Retrieved November 10, 2020, from https://archive.ics.uci.edu/ml/datasets/default_of_credit_card_clients

08 Appendix

APPENDIX A: DATA SET VARIABLE DESCRIPTION

The 25 variables in the data set are as follows:

1. ID: ID of each client
2. LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit
3. SEX: Gender (1=male, 2=female)
4. EDUCATION: (0=others, 1=graduate school, 2=university, 3=high school, 4=others, 5=others, 6=others)
5. MARRIAGE: Marital status (0=others, 1=married, 2=single, 3=divorced)
6. AGE: Age in years
7. PAY_0: Repayment status in September, 2005 (-2=no consumption, -1=pay duly, 0=use of revolving credit, 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)
8. PAY_2: Repayment status in August, 2005 (scale same as above)
9. PAY_3: Repayment status in July, 2005 (scale same as above)
10. PAY_4: Repayment status in June, 2005 (scale same as above)
11. PAY_5: Repayment status in May, 2005 (scale same as above)
12. PAY_6: Repayment status in April, 2005 (scale same as above)
13. BILL AMT1: Amount of bill statement in September, 2005 (NT dollar)
14. BILL AMT2: Amount of bill statement in August, 2005 (NT dollar)
15. BILL AMT3: Amount of bill statement in July, 2005 (NT dollar)
16. BILL AMT4: Amount of bill statement in June, 2005 (NT dollar)
17. BILL AMT5: Amount of bill statement in May, 2005 (NT dollar)
18. BILL AMT6: Amount of bill statement in April, 2005 (NT dollar)
19. PAY AMT1: Amount of previous payment in September, 2005 (NT dollar)
20. PAY AMT2: Amount of previous payment in August, 2005 (NT dollar)
21. PAY AMT3: Amount of previous payment in July, 2005 (NT dollar)
22. PAY AMT4: Amount of previous payment in June, 2005 (NT dollar)
23. PAY AMT5: Amount of previous payment in May, 2005 (NT dollar)
24. PAY AMT6: Amount of previous payment in April, 2005 (NT dollar)
25. default.payment.next.month: Default payment (1=yes, 0=no)

APPENDIX B: Model Results with Synthetic Minority Over-sampling Technique (SMOTE)

Model	Test MSE	Accuracy	ROC
Logistic Regression	0.172	0.823	0.714
Logistic Regression(LASSO)	0.172	0.822	0.713
Logistic Regression(Rpart)	0.173	0.825	0.699
PCR (1 sigma)	0.179	0.811	0.699
PCR (min CV)	0.261	0.819	0.705
PCR w/ LASSO (λ = 1SE)	0.177	0.818	0.706
PLS (1SE)	0.177	0.816	0.709
PLS (min CV)	0.177	0.819	0.708
Decision Tree (Anova)	0.162	0.817	0.742
Bagged Trees (ntree = 5000, mtry = 23)	0.133	0.826	0.748
Random Forest (ntree = 5000, mtry = 7)	0.133	0.826	0.769
GBM (CV)	**0.132**	**0.826**	**0.776**
GBM (OOB)	**0.133**	**0.827**	**0.771**
Neural Network (w/o Feature Selection)	0.178	0.822	0.762
Neural Network (w/ Feature Selection)	0.177	0.823	0.761