Credit Risk Modelling

Model Report

Supervisor: Mr. Ashley Albern Fernandez
Mentor: Mr. Ethan Jeremiah Chity
Department: Data Science and Engineering

Name: Jo Wayne Tan
Date: 01/10/2019

AI & Advanced Analytics
Menara Maxis
Kuala Lumpur
Malaysia

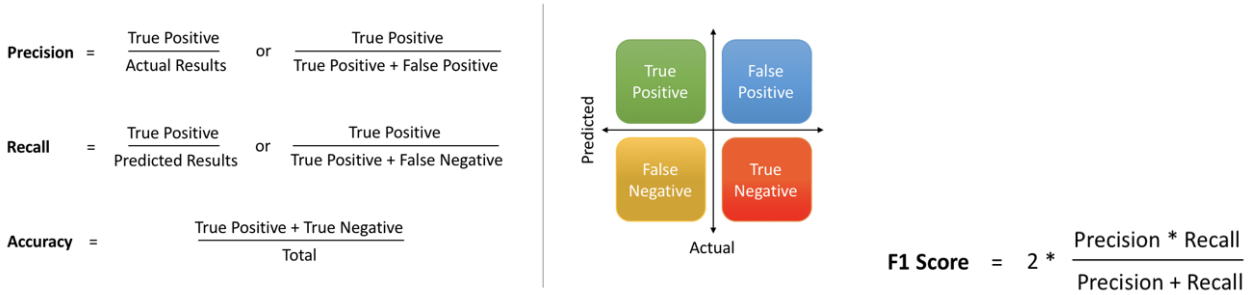
Abstract

Due to cost considerations, this project attempts to build an efficient tool for the Consumer Business department, in order to easily and accurately assess the default risk of our clients if offered a contract device. Precisely, the main purpose of this project is to predict the credit worthiness of consumers by scoring them, thus it is a multi-classification problem. The dataset from Kaggle consists of roughly 30,000 consumers characterized by 25 variables. The models implemented present an average predictive power (mean AUC around 0.717): they are obtained by using logistic regression, random forest and neural network techniques.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Mr. Ashley Albern Fernandez, and my mentor/buddy, Mr. Ethan Jeremiah Chitty, for providing their invaluable guidance, comments and suggestions throughout the course of my internship. They are always patient to help me out with questions regarding machine learning and programming. I am grateful to my friends, Shaun Fendi Gan and Kai Hui Wu, for making my time in Maxis joyful but always efficient.

Glossary



(a) **(b)**

Figure 1: Images taken from [1].

From Figure 1:
Accuracy - a measure of how often the model is correct.

Confusion Matrix - a tool used in machine learning to help visualise the performance of an algorithm in tabular form, shown on the right of Figure 1(a).

F1 score - a single metric that combines recall and precision using the harmonic mean [2].

Precision - the percentage of results which are relevant [1].

Recall - also known as True positive rate or Sensitivity, is the percentage of total relevant results correctly classified by the algorithm [1]. Precision and recall are essentially a trade-off.

Others:
Batch Size - total number of training examples present in a single batch.

Classification - predicting which class an item belongs to. Some classifiers are binary, resulting in a yes/no decision. Others are multi-class, able to categorize an item into one of several categories [3].

Cross Entropy - a measure of the difference between two probability distributions for a given random variable or set of events [4]:

$$H(P, Q) = -\sum_{i} P_i \log_2(Q_i) \tag{1}$$

where p is the true distribution and q is the predicted distribution.

Entropy - a measure of impurity, disorder or uncertainty in a bunch of samples:

$$H(P) = -\sum_i P_i \log_2(P_i) \tag{2}$$

where p is the underlying probability distribution.

Epoch - an entire data set passed forward and backward through the model once. Weights are updated in this process.

False negative - also known as Type II Error, is the error of not rejecting a null hypothesis when the alternative hypothesis is the true state of nature [5].

False positive - also known as Type I Error, is the error of rejecting a null hypothesis when it is actually true [5].

False positive rate - also known as Fall-out, is the proportion of the units with a known negative condition for which the predicted condition is positive [6]:

$$\frac{FalsePositive}{FalsePositive + TrueNegative} \tag{3}$$

Iterations - number of batches needed to complete 1 epoch.

Loss function - also known as cross-entropy loss, measures the performance of a classification problem whose outcome is a probability between 0 and 1. To optimize a model, minimize this so the predicted value converges to the actual value.

Mean Squared Error (MSE)- measures the average squared difference between the estimated values and the actual value:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{4}$$

Overfitting - when a model fits too closely to a training set. It learns and describes the noise in a training data instead of actual relationships between variables in a data. This is often a symptom of a too complex model, i.e., too many features or variables compared to number of observations.

True negative rate - also known as Specificity, is the proportion of the units with a known negative condition for which the predicted condition is negative [6]:

$$\frac{TrueNegative}{FalsePositive + TrueNegative} \tag{5}$$

Underfitting - when a model does not fit a training data and miss the trends in a data, e.g. fitting a linear model to a non-linear data. It cannot be generalized to new data. This is often a symptom of a very simple model, i.e., not enough predictors or independent variables.

Validation data - provide unbiased evaluation of a model's fitness. If the error is high, the model is overfitting.

Weights - the strength of connections between neurons in a neural network.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Business Requirements

Maximizing the Average Revenue per User (ARPU) is every telco's dream, to test a customer's limits, stretching their spending in every possible way. However, deal offerings have their own risks. This model will be used to offer contract devices to Postpaid Flex subscribers based on their given creditworthiness, and further modified to score Prepaid customers. The task is to build a homogeneous Credit Risk Model in house for the Consumer Business department instead of relying on credit services/agencies like CTOS.

Table 1.1: Different subscriber bases.

Prepaid	Flex	Postpaid
• Upsell • Capture new customers (sign contracts) and embark them on the Maxis journey • Cheapest deal out there win customers • Makes up most of the market, a huge potential source of revenue	• Upsell • Offer devices to put them on contract, increase stability • Monthly payment with a plan but without a contract, free to leave anytime	• Cross sell • Offer devices & maxis fibre • At the ending of the Maxis journey

1.2 Business Objectives

Business Objectives and Internal Goals:

1. Current model categorize customers as low, medium or high risk. Further classifying customers to give more granularity can create new possibilities to do more.

2. Model could be provided as a new business service to give micro loans to customers based on their credit scores on a web application/API.

3. Cold start modelling to deal with the lack of customer character data, for example billing info. Explore Social Network Analysis and Hidden Markov Model using personal info.

Chapter 2

Solutions Design

2.1 5 C's of Credit Risk

2.1.1 Building Blocks

1. Character ⇒ Customer stability, Loan history
2. Capacity ⇒ Monthly Debt to Income Ratio
3. Capital ⇒ Debt to Equity Ratio, contribution towards capital, seriousness
4. Conditions ⇒ Industry trends, Inflation etc
5. Collateral ⇒ Liens

2.1.2 Data Availability / Required

Primary focus is mostly on customer character and capacity.

Table 2.1: Data according to Building Blocks.

Character	Capacity	Capital	Conditions	Collateral
Payment delinquency data (amount & duration owed)	Spending power (maxis subscription data, browsing history)	Amount of money invested in business	Competitive Landscape	Real estates
Account history (suspended/ barred)	Account info (principal/sub)	Amount of down payment for a house	Supplier and Customer relations	
Internal delinquency data (billing info)	Home Location (iProperty)		Industry specific issues	
Personal info (age, address, income)	Entertainment (nexflix etc)			
Current Address Length	Call records (debt collection calls received)			
Employment Title	Transportation (car owner, public transport)			
Employment Length	Insurance plans			
Bankruptcy info				

Data Available	Data Unavailable

Chapter 3

Exploratory Data Analysis

3.1 Summary

Exploratory Data Analysis is an approach to gauge every nuance from a data at early encounter. It refers to the investigating measures on data to discover patterns, spot anomalies, test hypothesis and check assumptions with the help of summary statistics and graphical representations [7].

An EDA is performed on Default Credit Card of Taiwan Clients in 2005 [8] to get familiar with the available data. Understanding the core business objective and requirements are important before narrating them to a data requirement. EDA enables us to have a clear understanding on our dataset, and most importantly, validating the richness of the data in relation to the initial business objective and requirements.

```
df.describe(include='all')
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500	-0.016700
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	9.217904	1.123802
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000

Figure 3.1: Data set description.

The record count shown on the **count** row matches with the row number from the df.shape command, which is 30,000. All columns have the same count which means there are no missing fields.

```
df.nunique()
```

ID	30000
LIMIT_BAL	81
SEX	2
EDUCATION	7
MARRIAGE	4
AGE	56
PAY_0	11
PAY_2	11
PAY_3	11
PAY_4	11
PAY_5	10
PAY_6	10
BILL_AMT1	22723
BILL_AMT2	22346
BILL_AMT3	22026
BILL_AMT4	21548
BILL_AMT5	21010
BILL_AMT6	20604
PAY_AMT1	7943
PAY_AMT2	7899
PAY_AMT3	7518
PAY_AMT4	6937
PAY_AMT5	6897
PAY_AMT6	6939
default.payment.next.month	2

dtype: int64

(a) Number of unique values in each column.

```
df.head()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1
0	1	20000.0	2	2	1	24	2	2	-1	-1	...	0.0	0.0	0.0	0.0
1	2	120000.0	2	2	2	26	-1	2	0	0	...	3272.0	3455.0	3261.0	0.0
2	3	90000.0	2	2	2	34	0	0	0	0	...	14331.0	14948.0	15549.0	1518.0
3	4	50000.0	2	2	1	37	0	0	0	0	...	28314.0	28959.0	29547.0	2000.0
4	5	50000.0	1	2	1	57	-1	0	-1	0	...	20940.0	19146.0	19131.0	2000.0

(b) First 5 rows of the dataset.

Figure 3.2

The dataset contains 30000 rows and 25 columns with 15 numerical and 10 multi-categorical columns. The MARRIAGE column contains 0 and 3 (others) values which we will discuss later.

3.2 Categorical Columns

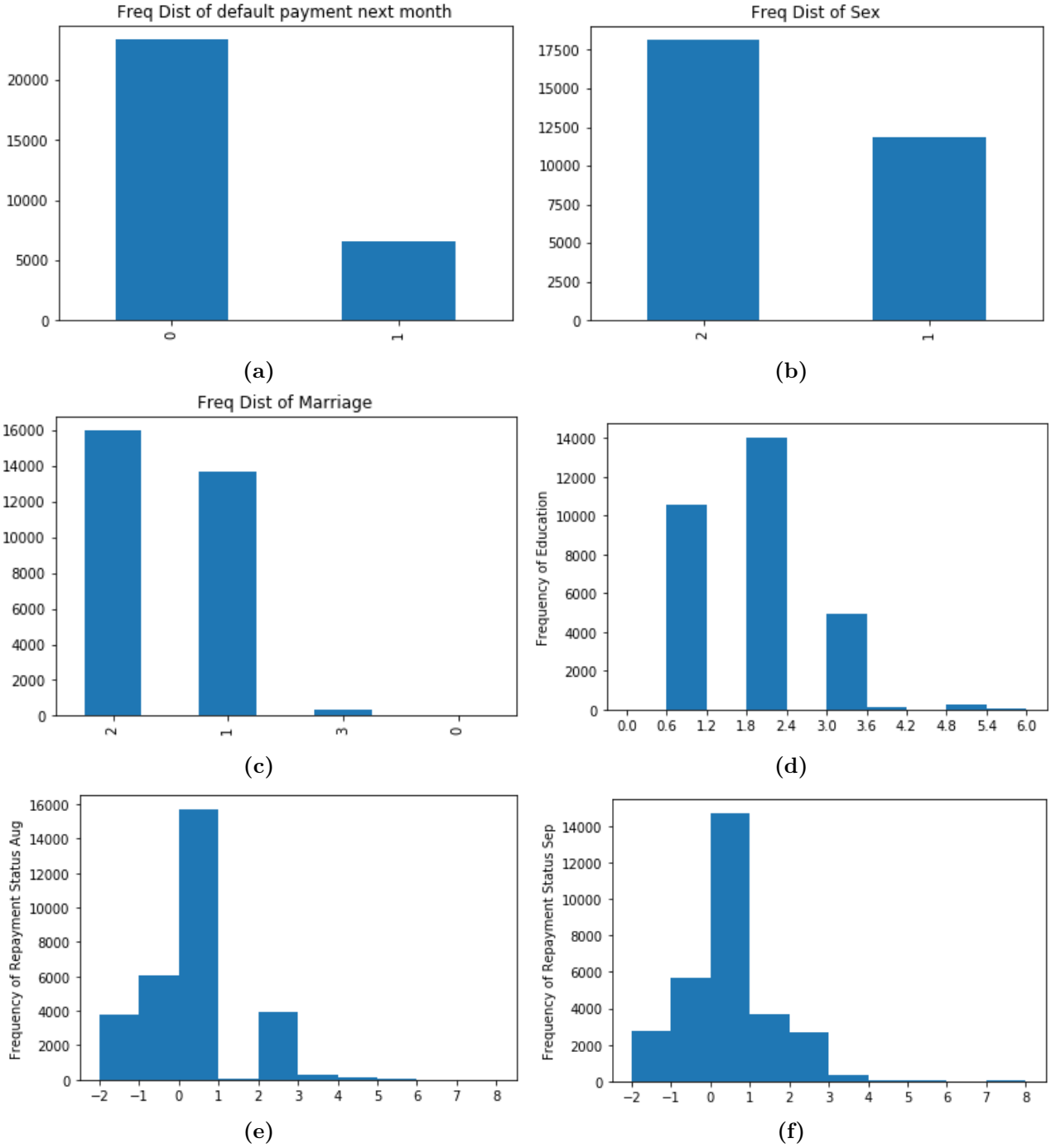
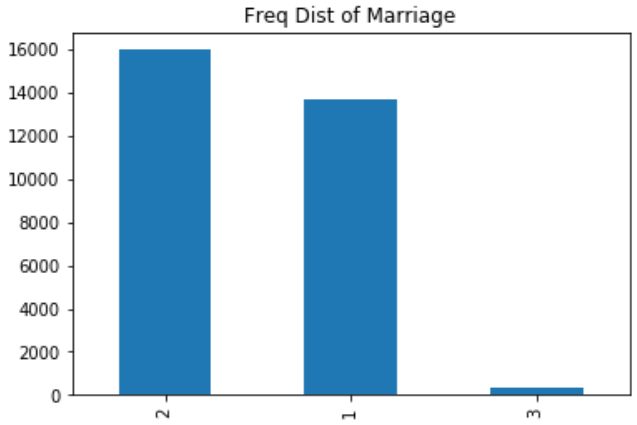


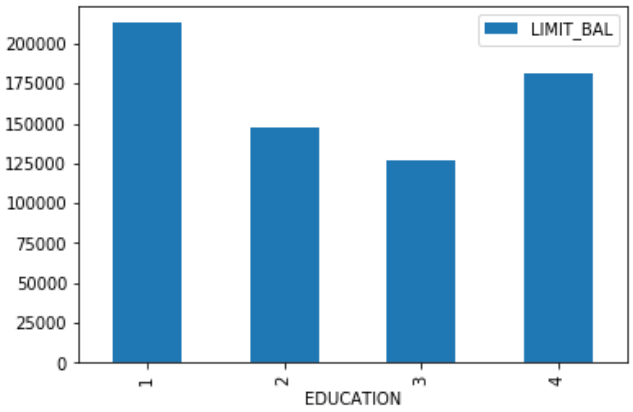
Figure 3.3: Bar plots.

Default payment (Figure 3.3(a)) is 22% of the total observations, balancing of dataset is required for an even amount of yes and no values. Unknown labels are present in the Marriage (0), EDUCATION (5 & 6) and PAY (-2 & 0) attributes. We have to find out what they actually mean or consider dropping them/merge with other labels.

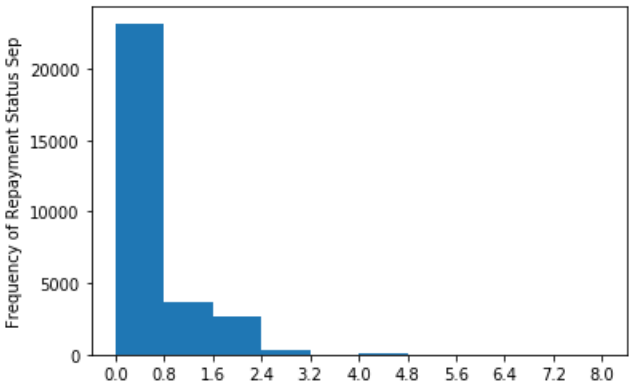
3.3 Fixing the Data Points



(a) Merged 0 to 3 (others).



(b) Merged 0, 4, 5, 6 to 4 (others).



(c) Merged -2, -1 to 0 (pay duly) for all PAY attributes.

Figure 3.4

3.4 Missing Value Analysis

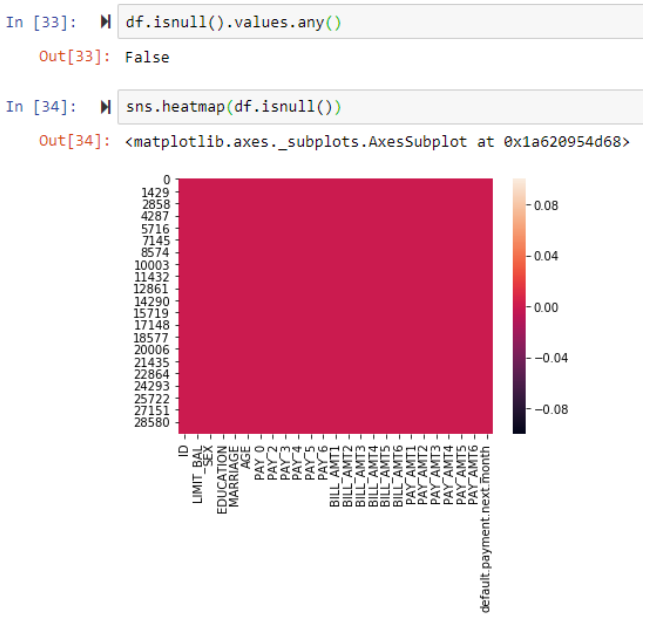


Figure 3.5: No null values detected in the dataset. A heatmap of the output of isnull() command is generated to verify the result.

3.5 Numerical Columns

(a) Univariate Distribution.

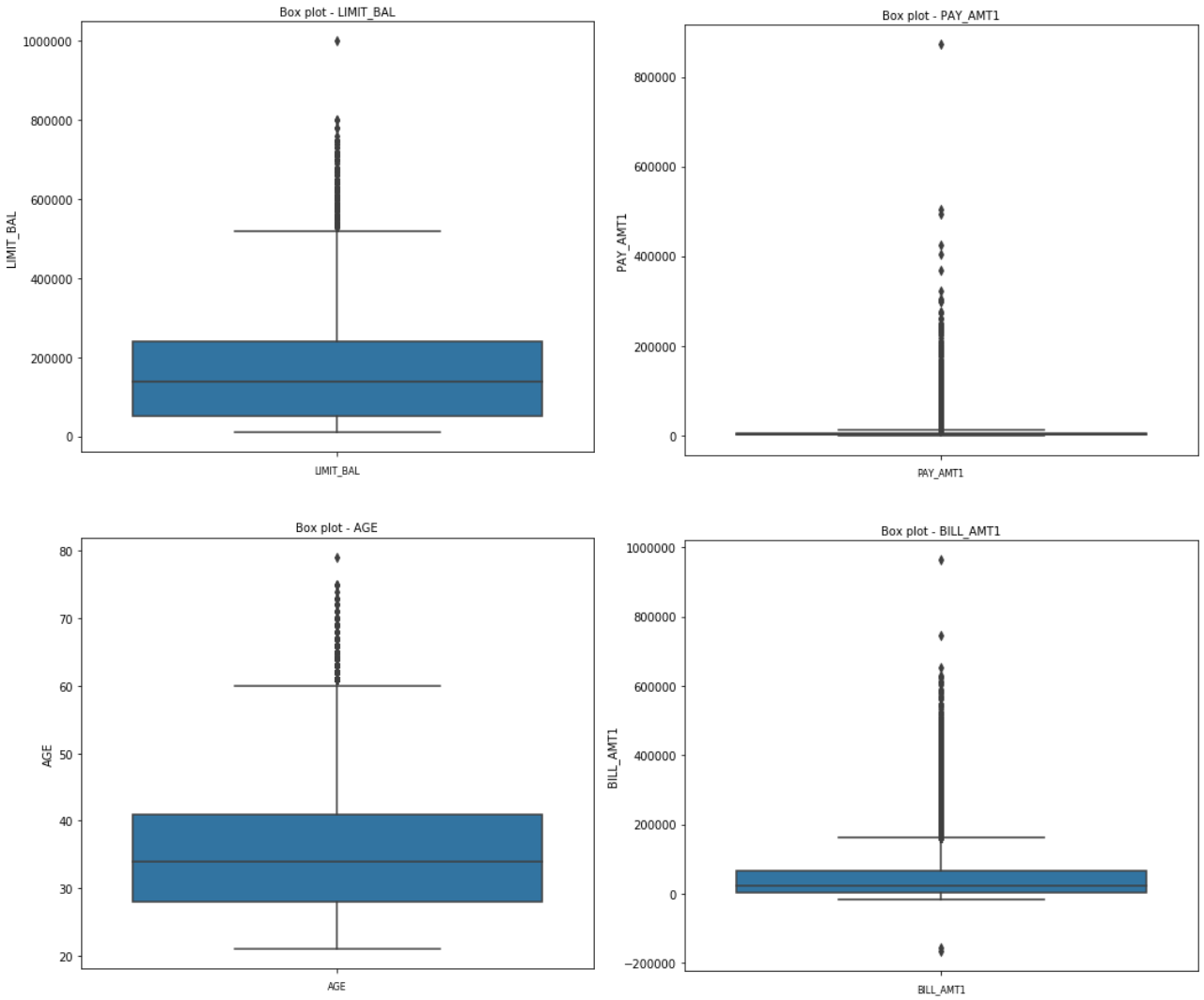
(b) Bivariate Distribution.

Figure 3.6

In Figure 3.5(a), excluding the LIMIT_BAL and AGE column, every other column is skewed to the left which indicates most of the values lie in the lower range.

While in Figure 3.5(b), the BILL_AMT1 and BILL_AMT2 scatter plots follow a linear pattern with an increasing slope. It is possible that all BILL_AMT columns follow a similar trend and we can consider only selecting a few of them. This linear pattern is not found in other scatter plots, therefore cannot draw any other conclusions.

3.6 Outlier Detection Analysis

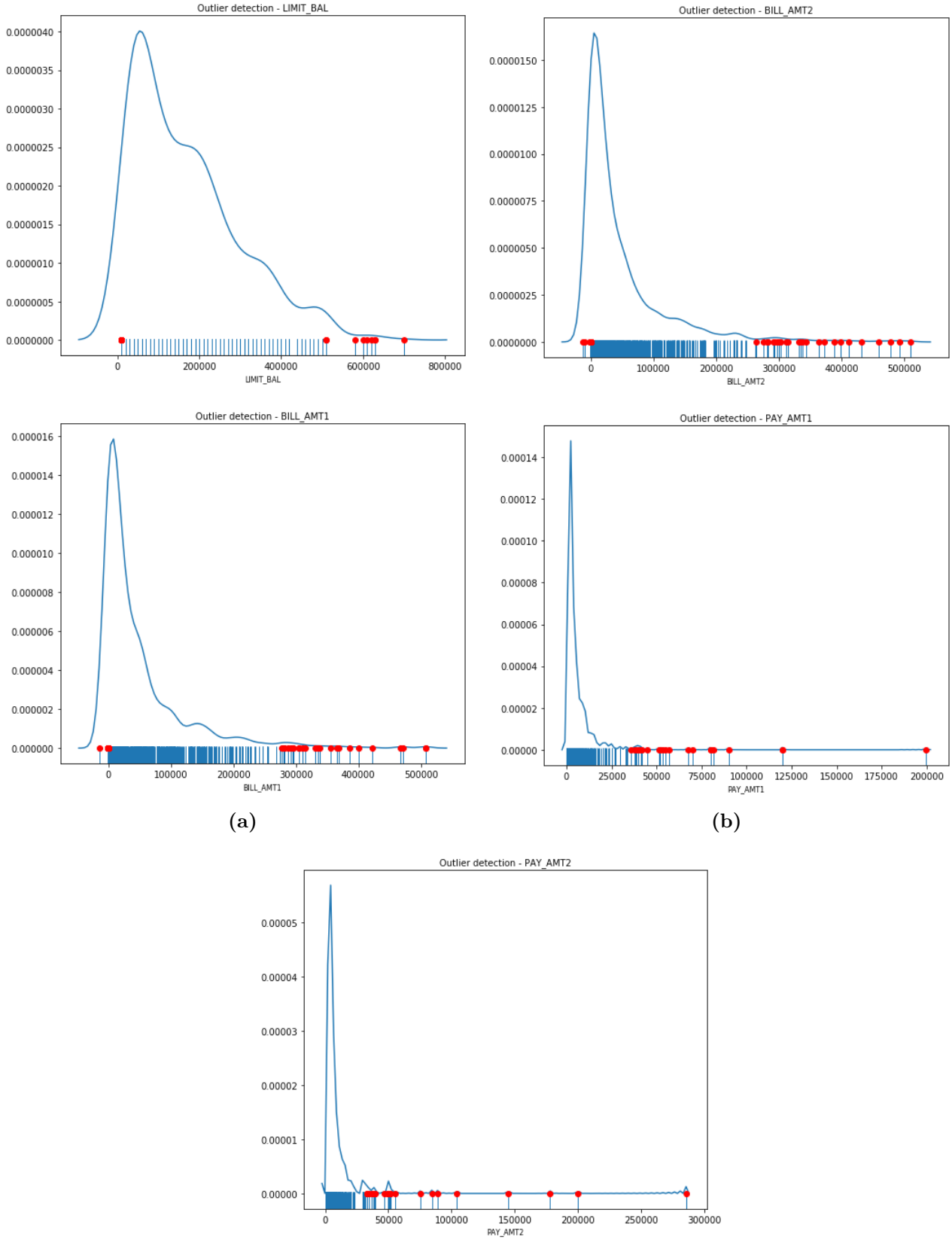


(a) **(b)**

Figure 3.7: Box Plots.

Outliers are observations that fall below Q1 – 1.5(IQR) or above Q3 + 1.5(IQR). Based on this definition the black dots are outliers in the attributes and the blue colored box is the IQR range. Although AGE is a numerical attribute, it makes sense to have people over 60 years old, so we assume there is no outlier in the AGE column and retain all values.

3.7 Percentile Based Outlier Removal



(a) **(b)**

(c)

Figure 3.8: Outlier detection plots.

Filter out outliers based on fixed percentile values. The values marked with a dot below in the x-axis of the graphs are removed from the column based on the default value threshold percentile (95%).

3.8 Correlation Matrix

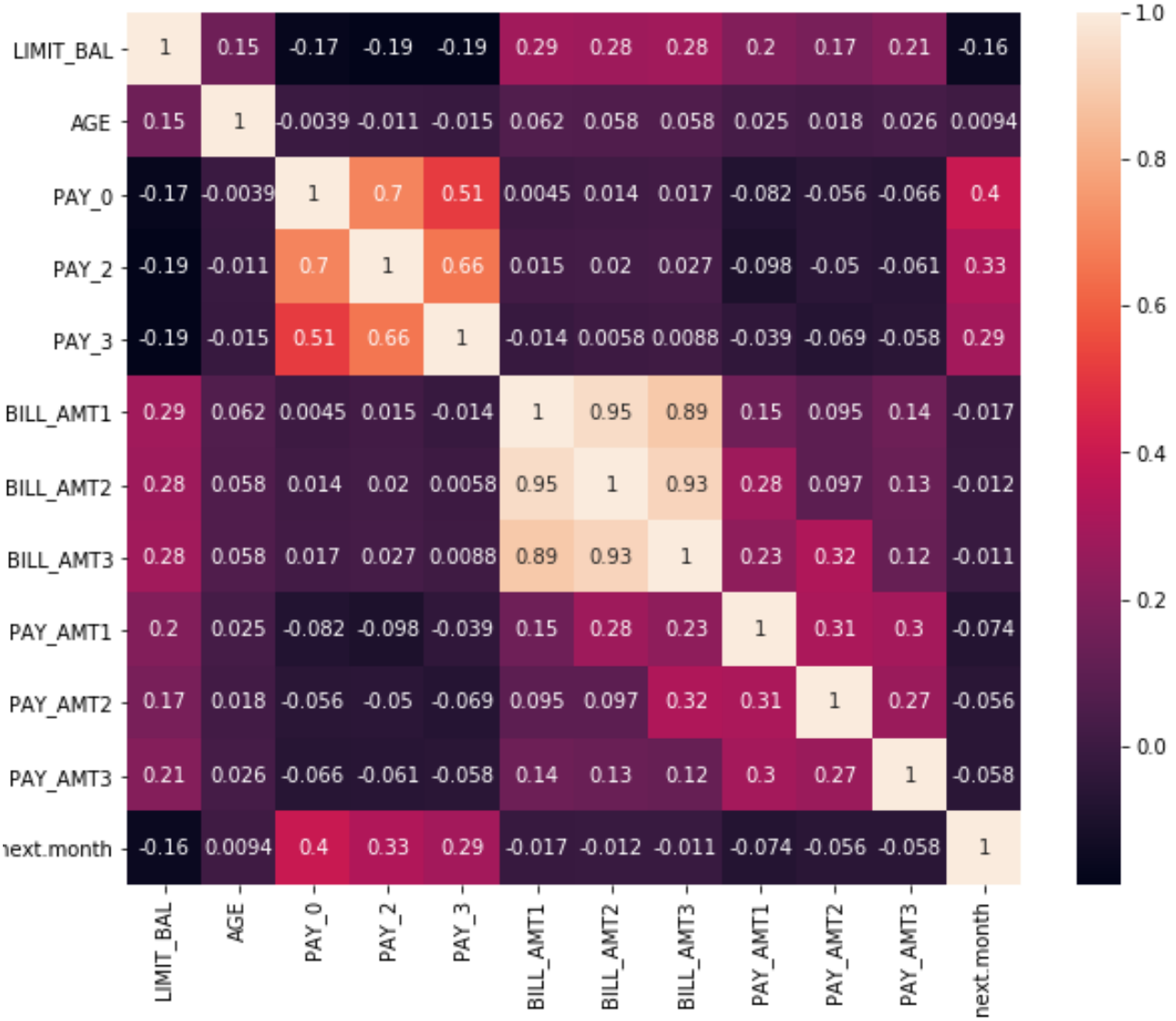
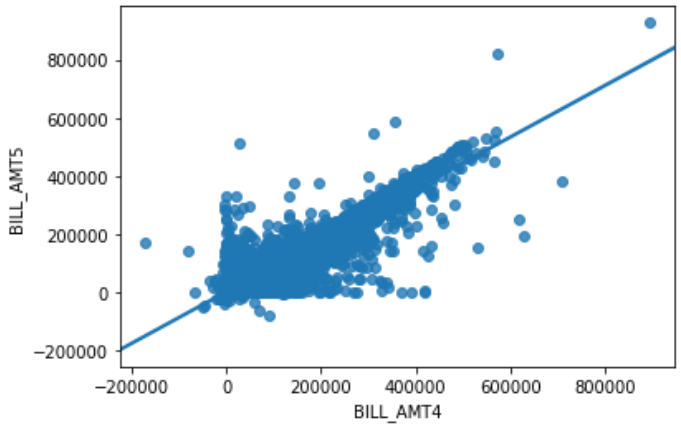


Figure 3.9: The correlation matrix.

Pairs which are highly correlated represent the same variance of the data set thus can be further analyzed to understand which attribute among the pairs are most significant for building the model. In this data set, the PAY and BILL_AMT attributes seem to be highly correlated. Otherwise all other numerical attributes are important and needs to be considered for building the model. PAY_0 seems to be the highest correlated attribute with Default payment.

3.9 Verifying Relationships

```
In [58]: sns.regplot(x='BILL_AMT4',y='BILL_AMT5',data=df)
Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x1a623df7ac8>
```




```
In [59]: sns.regplot(x='PAY_4',y='PAY_5',data=df)
Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x1a65be20eb8>
```

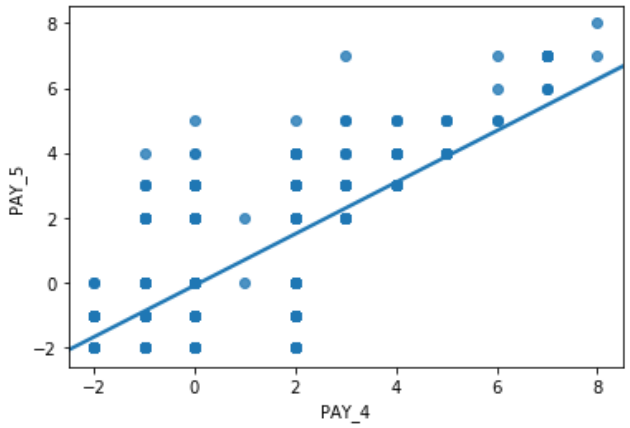


Figure 3.10: Regression plots.

Two different columns are used to verify the results from the matrix. These plots show the positive correlation within BILL_AMT attributes and PAY_ attributes.

3.10 Feature Engineering

1. Adding
   ⇒ Another interesting feature to consider is the Balance-to-Limit Ratio, which is the Amount of Credit being used (LIMIT_BAL) over the Total Credit Available to a client.

2. Removing
   ⇒ Remove unknown labels and select features with the strongest influence from highly correlated attributes to reduce dimensionality and minimize amount of info used on a model.

3. Changing
   ⇒ Convert multi-categorical attributes to numerical in order to prevent bias.

Chapter 4

Classification Model Development

4.1 Machine Learning Theory

4.1.1 Logistic Regression

Logistic Regression predicts the probability of a categorical binary dependent variable, i.e., predicts $P(Y = 1)$ as a function of X [9]. The Maximum Likelihood Estimation (MLE) algorithm determines the regression coefficient for the model that accurately predicts the probability of the binary dependent variable [9]. The logistic function (4.1) demonstrates the probability of dependent variable by independent factors [9], an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits [10]. The logit function is:

$$P = \frac{e^{log(odds)}}{1 + e^{log(odds)}} \tag{4.1}$$

where P is the probability of success and odds is the odd ratio ($\frac{p}{1-p}$) [9, 11].

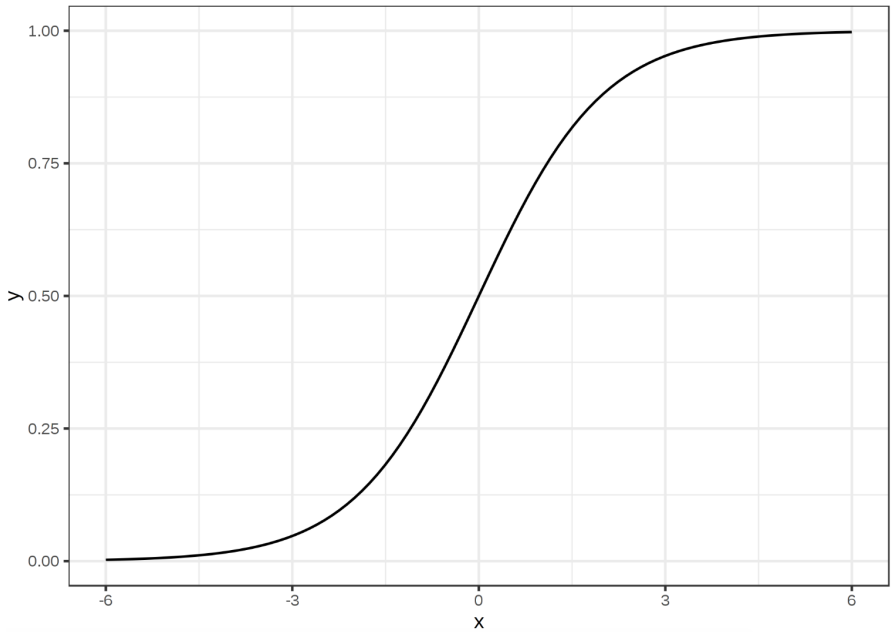


Figure 4.1: The logistic function. It outputs numbers between 0 and 1. At input 0, it outputs 0.5 [12].

4.1.2 Random Forest

Random forests are an example of an ensemble learner built on decision trees [13]. The model addresses overfitting by searching over a random N subset of the original features when constructing a tree to find the best decision trees [3, 14]. Different decision trees have different features, so these trees will be of different size, with different branches, making different predictions [14]. All the single trees are independent of each other, therefore by averaging out the impact of several uncorrelated decision trees, random forests tend to improve prediction of the final result. [14, 15]. This technique is called bagging, or bootstrap aggregation [14].

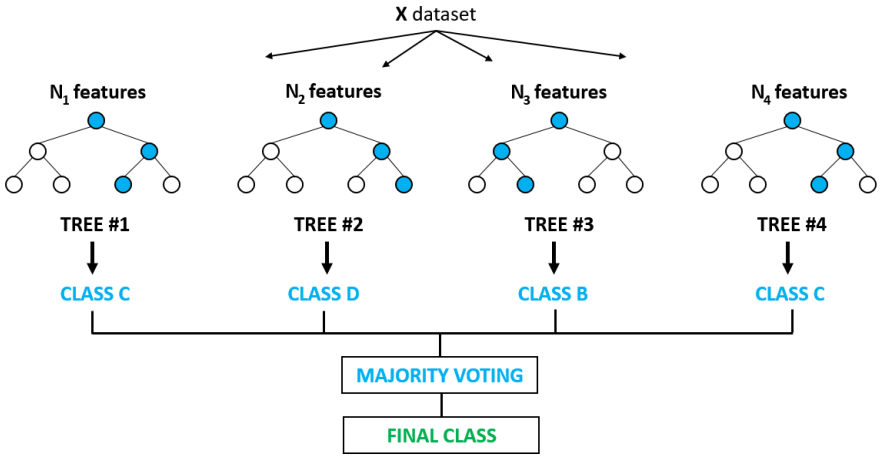


Figure 4.2: Image taken from [14].

4.1.3 Neural Networks

Artificial neural networks are built of simple elements called neurons, each taking a set of numeric values as input and maps them to a single output value [3, 16]. A neuron is a simple multi-input linear regression function, where the output is multiplied by a weight and passed through a non-linear activation function [3, 16].

A back-propagation neural network feed the calculated error rates back to the hidden layers, update the processing at each neuron, and adjust the weights to reduce the error rate. This supervised learning method requires a labelled data set [16].

The training starts by assuming random weights to each of the connections in the network. The algorithm then iteratively update the weights by showing training data to the network and updating the weights until the network is optimized. The network learns to improve its outputs by refining the link weights. A low weight suppresses a signal while a high weight amplifies it, therefore connections are de-emphasised if they are not needed to avoid overfitting [16, 17].

The activation functions used in the following neural network classifiers are the ReLU and Sigmoid function.

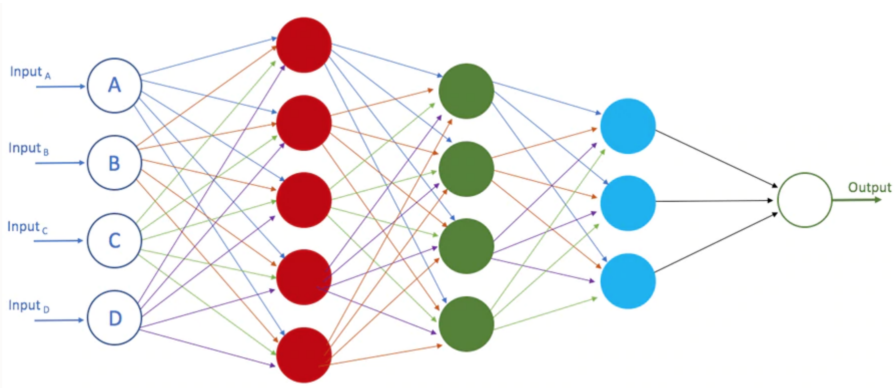


Figure 4.3: All the inputs are connected to a neuron in the hidden layer (red circles). By constructing multiple layers of neurons, each of which receives part of the input variables, and then passes on its results to the next layers, the network can learn very complex functions [3, 16].

4.1.4 ROC Curves

A receiver operating characteristic (ROC) curve plots the true positive rate (sensitivity) against the false positive rate (1 – specificity) for all possible cut off values. The true positive rate and false positive rate are calculated for a given table, based on a single cut off value [6].

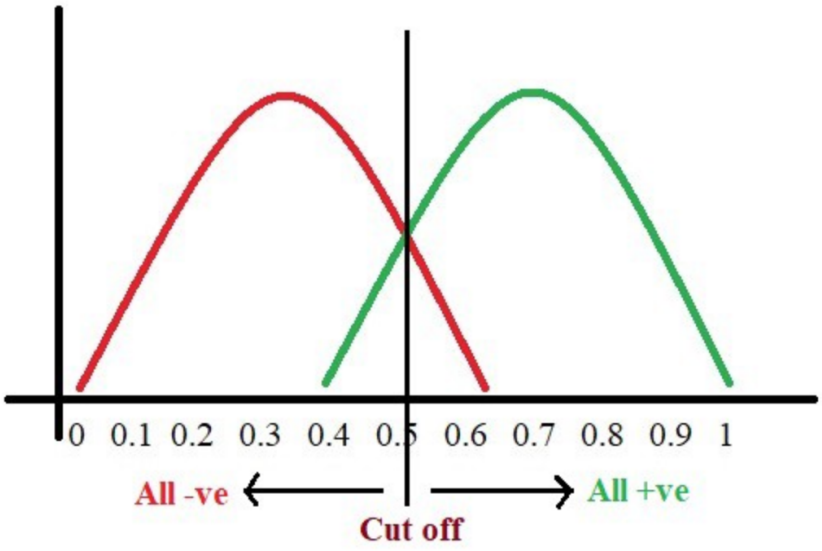


Figure 4.4: A probability distribution plot taken from [18].

The cut off value, also known as the threshold, of a probability distribution plot is normally 0.5 on default. Choosing a classification threshold is a business decision: to minimize false positive rate or maximize true positive rate. The lower the threshold, the more predicted positive values, the higher the sensitivity.

Each point on the ROC curve represents a different cut off value. As the true positive rate increases, the false positive rate increases. The better the diagnostic test, the more quickly the true positive rate climbs to 1 (or 100%) [6].

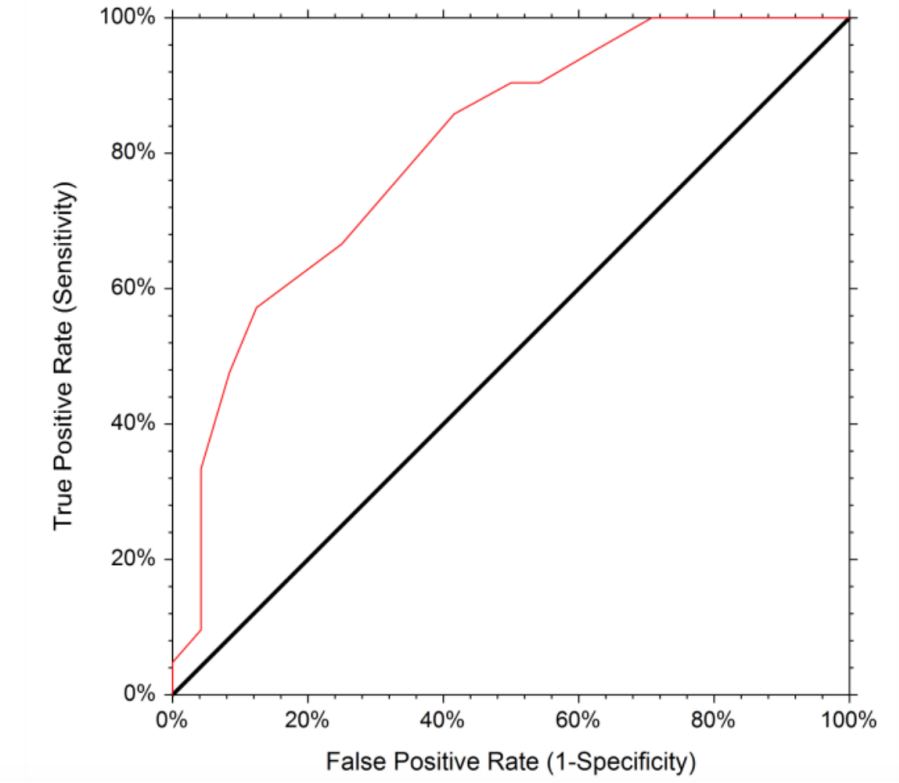


Figure 4.5: Example of a ROC curve taken from [6].

The red curve in Figure 4.5 is the curve of probability and the black diagonal line represents the ROC curve of a purely random classifier (area = 0.5). A good classifier would have a ROC curve that is almost vertical from (0,0) to (0,1) and then horizontal to (1,1) [6], staying as far away from the reference line as possible, towards the top-left corner (area $\approx$ 1).

AUC ROC Curve

The area under a receiver operating characteristic curve is a common tool used to evaluate the performance of a binary/multi-class classifiers, defined as the average true positive rate across all possible false positive rates [6]. It tells how good a model is capable of distinguishing between classes. The greater the area, the better the prediction (0s as 0s and 1s as 1s).

The possible values of AUC range from 0.5 (no diagnostic ability) to 1.0 (perfect diagnostic ability and separability). The physical interpretation of the AUC is the probability that the criterion value of an individual drawn at random from the population of those with a positive condition is larger than the criterion value of another individual drawn at random from the population of those where the condition is negative [6]. It indicates how well the probabilities from the positive classes are separated from the negative classes [18].

4.2 Training Complexity

The data set is used to train 3 different machine learning models.

1. Logistic Regression
2. Random Forest
3. Neural Network

But first, it has to go through the following processes before feeding to any model:

1. One-hot encoding
2. Balancing data distribution
3. Splitting data set into training and testing set

4.2.1 Creating Dummy Variables

```
df1.columns.values
```

```
array(['LIMIT_BAL', 'SEX', 'AGE', 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3',
       'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1', 'PAY_AMT2',
       'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
       'default.payment.next.month', 'EDUCATION_1', 'EDUCATION_2',
       'EDUCATION_3', 'EDUCATION_4', 'MARRIAGE_1', 'MARRIAGE_2',
       'MARRIAGE_3', 'PAY_0_0', 'PAY_0_1', 'PAY_0_2', 'PAY_0_3',
       'PAY_0_4', 'PAY_0_5', 'PAY_0_6', 'PAY_0_7', 'PAY_0_8', 'PAY_2_0',
       'PAY_2_1', 'PAY_2_2', 'PAY_2_3', 'PAY_2_4', 'PAY_2_5', 'PAY_2_6',
       'PAY_2_7', 'PAY_2_8', 'PAY_3_0', 'PAY_3_1', 'PAY_3_2', 'PAY_3_3',
       'PAY_3_4', 'PAY_3_5', 'PAY_3_6', 'PAY_3_7', 'PAY_3_8', 'PAY_4_0',
       'PAY_4_1', 'PAY_4_2', 'PAY_4_3', 'PAY_4_4', 'PAY_4_5', 'PAY_4_6',
       'PAY_4_7', 'PAY_4_8', 'PAY_5_0', 'PAY_5_2', 'PAY_5_3', 'PAY_5_4',
       'PAY_5_5', 'PAY_5_6', 'PAY_5_7', 'PAY_5_8', 'PAY_6_0', 'PAY_6_2',
       'PAY_6_3', 'PAY_6_4', 'PAY_6_5', 'PAY_6_6', 'PAY_6_7', 'PAY_6_8'],
```

Figure 4.6: Column variables after one-hot encoding.

Multi-categorical variables are converted into separate columns for each category, containing only values of 0s and 1s in each column. Most ML models cannot deal with categorical variables. This form could be provided to ML algorithms to do a better job in prediction.

4.2.2 Handling Imbalance Data Distribution

```
count_no_default = len(df[df['default.payment.next.month']==0])
count_default = len(df[df['default.payment.next.month']==1])
pct_of_no_default = count_no_default / (count_no_default + count_default)
print("percentage of no default is", pct_of_no_default*100)
pct_of_default = count_default / (count_no_default + count_default)
print("percentage of default", pct_of_default*100)
```

```
percentage of no default is 77.81148611637803
percentage of default 22.18851388362196
```

(a)

```
print(len(df1.loc[df1['default.payment.next.month'] == 1]))
print(len(df1.loc[df1['default.payment.next.month'] == 0]))
```

```
5266
18467
```

(b)

Figure 4.7: Showing percentage of defaulters and non-defaulters.

Amount of yes and no values are uneven. The limiting value is yes (default), with 5266 observations.

4.2.3 Split Ratio

```
# Create Balanced Training Dataset
train = df1.loc[df1['default.payment.next.month'] == 1].sample(n=3680,random_state=7)
train = train.append(df1.loc[df1['default.payment.next.month'] == 0].sample(n=3680,random_state=7),sort=False)
train = train.sample(frac=1, random_state=7)
test = df1.drop(train.index)
```

```
# Creating input features and target variables
X_train = train.loc[:, train.columns != 'default.payment.next.month']
y_train = train.loc[:, train.columns == 'default.payment.next.month']
X_test = test.loc[:, test.columns != 'default.payment.next.month']
y_test = test.loc[:, test.columns == 'default.payment.next.month']
```

Figure 4.8: Splitting the data set into training and testing data.

The split ratio used is 7:3 train to test. 70% of 5266 is about 3680, so 3680 number of yes and no observations are taken from the data set as the training set. All remaining observations are used as the testing set. The train_test_split function could also be used to split the data set.

4.3 Testing & Results

Logistic Regression

Recursive Feature Elimination

RFE repeatedly construct a model and choose either the best or worst performing feature, setting the feature aside and repeating the process with the rest of the features. This process is applied until all features in the data set are exhausted. RFE select features by recursively considering smaller and smaller sets of features.

Results: Logit

Model:	Logit	Pseudo R-squared:	0.117
Dependent Variable:	default.payment.next.month	AIC:	12761.4286
Date:	2019-08-22 15:31	BIC:	12899.1703
No. Observations:	10400	Log-Likelihood:	-6361.7
Df Model:	18	LL-Null:	-7208.7
Df Residuals:	10381	LLR p-value:	0.0000
Converged:	0.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
EDUCATION_3	-0.4068	0.0531	-7.6549	0.0000	-0.5110	-0.3026
MARRIAGE_3	-0.3350	0.1832	-1.8285	0.0675	-0.6941	0.0241
PAY_0_1	0.2997	0.0689	4.3478	0.0000	0.1646	0.4348
PAY_0_2	1.7076	0.0907	18.8233	0.0000	1.5298	1.8854
PAY_0_3	1.4446	0.2572	5.6173	0.0000	0.9406	1.9487
PAY_0_8	0.1987	0.6060	0.3279	0.7430	-0.9891	1.3866
PAY_2_1	-0.1403	0.6889	-0.2037	0.8386	-1.4905	1.2099
PAY_2_2	0.3321	0.0881	3.7676	0.0002	0.1593	0.5049
PAY_2_3	1.0122	0.2459	4.1170	0.0000	0.5303	1.4941
PAY_2_4	0.1965	0.4250	0.4624	0.6438	-0.6364	1.0294
PAY_2_6	0.1096	1.5852	0.0692	0.9449	-2.9974	3.2166
PAY_3_1	-0.0107	1.5960	-0.0067	0.9946	-3.1387	3.1173
PAY_3_2	0.4573	0.0780	5.8639	0.0000	0.3044	0.6101
PAY_3_3	0.6513	0.3040	2.1425	0.0322	0.0555	1.2471
PAY_3_5	0.1041	1.4175	0.0734	0.9415	-2.6741	2.8823
PAY_4_8	0.0070	67847809.6429	0.0000	1.0000	-132979263.3231	132979263.3370
PAY_5_2	0.3859	0.0827	4.6675	0.0000	0.2238	0.5479
PAY_5_3	0.7098	0.3411	2.0812	0.0374	0.0413	1.3783
PAY_5_8	0.0070	67847809.6429	0.0000	1.0000	-132979263.3231	132979263.3370
PAY_6_5	0.0651	1.4156	0.0460	0.9633	-2.7094	2.8395

Figure 4.9: P-value Analysis

The p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low p-value indicates that you can reject the null hypothesis. In other words, a predictor that has a low p-value is likely to be a meaningful addition to your model because changes in the predictor's value are related to changes in the response variable. Variables with p-values larger than 0.05 are removed.

Results

```
# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter distribution using uniform distribution
C = uniform(loc=0, scale=4)

fit_intercept = [True, False]

# Create hyperparameter options
hyperparameters = dict(C=C, fit_intercept=fit_intercept, penalty=penalty)
```

```
# Create randomized search 5-fold cross validation and 100 iterations
clf = RandomizedSearchCV(logreg, hyperparameters, random_state=7, n_iter=100, cv=5, verbose=0, n_jobs=-1)
```

```
# Fit randomized search
best_model = clf.fit(X_train_lg, y_train_lg)
```

(a) Range of parameters are defined and a Randomized search on hyper parameters was performed to predict the parameters giving the best accuracy.

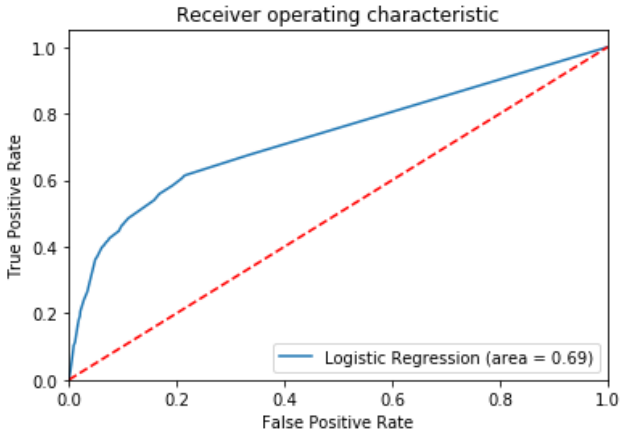
```
=== Confusion Matrix ===
[[12375  2412]
 [  716   870]]
```

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.95      0.84      0.89     14787
           1       0.27      0.55      0.36      1586

    accuracy                           0.81     16373
   macro avg       0.61      0.69      0.62     16373
weighted avg       0.88      0.81      0.84     16373
```

(b) Scores given to 1 (default) are comparably low.



(c) AUC ROC curve of the Logistic Regression model.

Figure 4.10

Random Forest Classifier

Feature Importance

```
''' Top 10 most important features for our current model '''
fi = rf_feat_importance(rfc, df_trn)
fi[:10]
```

	cols	imp
31	PAY_2_0	0.067939
2	AGE	0.060526
3	BILL_AMT1	0.059761
0	LIMIT_BAL	0.058318
4	BILL_AMT2	0.054321
10	PAY_AMT2	0.052711
9	PAY_AMT1	0.050030
8	BILL_AMT6	0.049386
5	BILL_AMT3	0.048737
12	PAY_AMT4	0.048288

Figure 4.11: Ranking the features according to their importance.

A random forest classifier was created using the raw training set to find out the importance of every feature. Features with an importance of smaller than 0.05 are removed from the data set to reduce the model training time and reduce overfitting.

Results

```
# number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]

# number of features at every split
max_features = ['auto', 'sqrt']

# max depth
max_depth = [int(x) for x in np.linspace(100, 500, num = 11)]
max_depth.append(None)

# minimum number of samples required to be at a leaf node
min_samples_leaf = [int(x) for x in np.linspace(1, 5, 5, endpoint=True)]
```

(a) Range of parameters are defined and a Randomized search on hyper parameters was performed to predict the parameters giving the best accuracy.

```
=== Confusion Matrix ===
[[11702  3085]
 [  578  1008]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.95      0.79      0.86     14787
           1       0.25      0.64      0.35      1586

    accuracy                           0.78     16373
   macro avg       0.60      0.71      0.61     16373
weighted avg       0.88      0.78      0.82     16373


=== All AUC Scores ===
[0.75542279 0.75504257 0.78911902 0.79788255 0.79552291]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.778597965977698
```

(b) Scores given to 1 (default) are relatively low.

Figure 4.12

Neural Network

Scaling & Principal Component Analysis

```
var_weight = pd.DataFrame(components_df1.sum(axis=0)).reset_index()
var_weight.rename(columns={'index':'feature',0:'variance_weight'},inplace=True)
var_weight.sort_values(by='variance_weight',ascending=False,inplace=True)
var_weight.head(12)
```

	feature	variance_weight
0	LIMIT_BAL	0.201704
10	PAY_AMT2	0.188626
9	PAY_AMT1	0.184079
5	BILL_AMT3	0.177200
4	BILL_AMT2	0.172779
14	PAY_AMT6	0.172419
6	BILL_AMT4	0.171049
7	BILL_AMT5	0.169622
13	PAY_AMT5	0.168853
8	BILL_AMT6	0.168538
11	PAY_AMT3	0.168120
3	BILL_AMT1	0.166425

Figure 4.13: Variance weight of top 12 features.

Numerical features are scaled to values of within -1 and 1. A common way of speeding up a ML algorithm is by using principal component analysis. PCA is a dimensionality reduction method, transforming a large set of variables into a smaller one that still contains most of its information. This preservation method slightly reduces accuracy but greatly improves simplicity which shortens model training time and reduces computational cost. PCA is used to reduce input dimensions by computing the variance weight of each feature and selecting the top 12 features that weigh the most as input features.

Results

```
def build_classifier(learn_rate=0.01, neurons=8):
    classifier = Sequential()
    #First Hidden Layer
    classifier.add(Dense(neurons, activation='relu', kernel_initializer='random_uniform', input_dim=42))
    #Second  Hidden Layer
    classifier.add(Dense(neurons, activation='relu', kernel_initializer='random_uniform'))
    #Output Layer
    classifier.add(Dense(1, activation='sigmoid', kernel_initializer='random_uniform'))
    #Compiling the neural network
    optimizer = optimizers.Adam(lr=learn_rate)
    classifier.compile(optimizer=optimizer, loss='binary_crossentropy', metrics =['accuracy'])
    return classifier
```

```
classifier = KerasClassifier(build_fn=build_classifier, verbose=0)
```

```
batch_size = [10, 20, 40, 60, 80, 100]
epochs = [10, 50, 100]
learn_rate = [0.001, 0.01, 0.1, 0.2, 0.3]
neurons = [1, 5, 10, 15, 20, 25, 30]
param_grid = dict(batch_size=batch_size, epochs=epochs, learn_rate=learn_rate, neurons=neurons)
```

```
grid = RandomizedSearchCV(estimator=classifier, param_distributions=param_grid, n_iter = 10, cv = 3, ran
grid_result = grid.fit(X_train_nn, y_train_nn)
```

(a) Range of parameters are defined and a Randomized search on hyper parameters is performed to predict the parameters giving the best accuracy.

```
=== Confusion Matrix ===
[[8309 6478]
 [ 332 1254]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.96      0.56      0.71     14787
           1       0.16      0.79      0.27      1586

    accuracy                           0.58     16373
   macro avg       0.56      0.68      0.49     16373
weighted avg       0.88      0.58      0.67     16373
```

(b) Model scores.

Figure 4.14: This model gave the worst accuracy, but the highest recall score.

4.4 Model Gaps & Limitations

- Generally, the models gave good accuracy because they performed well on the non-default (0) predictions.
- But, all of them scored badly on the default (1) predictions, which failed the objective of predicting clients who are going to default.
- This is due to the super low number of defaulters in the entire data set, therefore a classifier seems to have a decent accuracy by labelling most clients as non-defaulters.

Table 4.1: Model Gaps & Limitations

	Logistic Regression	**Random Forest**	**Neural Network**
Advantages	Does not require too many computational resources	Highly accurate classifier	Can create large combination of different structures based on: 1. No. of layers 2. Selection of activation function 3. Normalization layers
	Does not require scaling	Effective for estimating missing data and maintains accuracy	
	Easy to implement	Does not require scaling	
Disadvantages	Cannot solve non-linear problems	Hard to calibrate	Require huge amount of data
	Prone to overfitting	Prone to overfitting	Needs to fill missing values in a table Scaling required

Cross Validation

Cross validation is done to:

1. Reduce variability by performing multiple rounds of cross validation with different subsets and combine the results to estimate the predictive performance.
2. Prevent/minimize overfitting which affects predictability of model causing lower accuracy or ungeneralized predictions.
3. Ensure high accuracy of model is not by chance (seasonal, trends, skewed data fits).

K-fold

K-fold is a type of cross validation method that randomly divides a data set into k number of groups/folds of approximately equal size. The first/last fold is kept for testing while the model trains on k-1 folds. Accuracy is then determined by the average of MSE over k folds, so the results are not biased by any single partition:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{4.2}$$

```
cv = StratifiedKFold(n_splits=10, random_state=14, shuffle=True)
```

Figure 4.15: The stratified k-fold command.

The n_splits value above represents the number of k folds. 9 folds are trained while the remaining fold is used to validate the result, the process is repeated until each fold of the 10 folds have been used as the validation set.

4.5 Second Testing Iteration

The number of yes and no observations is decreased to 500 each, 1000 observations total in the testing set. All remaining observations are used as training set.

Number of k folds is 10, randomly picking 10 folds from the training set, which means the observations in each fold might overlap. This is to create randomness in the training data set to improve generalization.

Grid Search is also used instead of Randomized Search to generate every possible combination of parameters and pick the best performing ones. The parameters chosen are the ones that appeared the most frequent in the Randomized Search results.

Logistic Regression

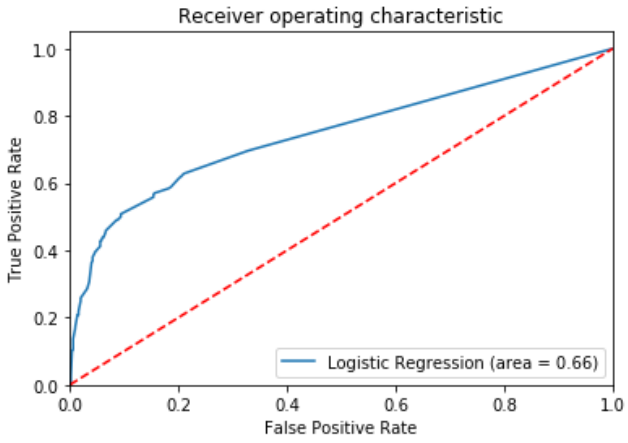
```
=== Confusion Matrix ===
[[479  21]
 [316 184]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.60      0.96      0.74       500
           1       0.90      0.37      0.52       500

    accuracy                           0.66      1000
   macro avg       0.75      0.66      0.63      1000
weighted avg       0.75      0.66      0.63      1000
```

(a)



(b)

Figure 4.16: Results.

Accuracy and every other score dropped besides recall of non-default (0) and precision and f1-score of default (1) prediction.

90% of all yes value observations are correctly identified, but the algorithm classified a lot more other observations as a yes value, which are incorrect. This is considered an improvement because the goal of the model is to identify default clients not non-default ones.

Random Forest

```
=== Confusion Matrix ===
[[477  23]
 [328 172]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.59      0.95      0.73       500
           1       0.88      0.34      0.49       500

    accuracy                           0.65      1000
   macro avg       0.74      0.65      0.61      1000
weighted avg       0.74      0.65      0.61      1000



=== All AUC Scores ===
[0.75609134 0.75422967 0.79031684 0.79828602 0.79695885]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.7791765448581272
```

Figure 4.17: Results.

Random forest gave a similar result as logistic regression, with a high recall and precision score for non-default and default predictions respectively.

Neural Network

```
=== Confusion Matrix ===
[[431  69]
 [357 143]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.55      0.86      0.67       500
           1       0.67      0.29      0.40       500

    accuracy                           0.57      1000
   macro avg       0.61      0.57      0.54      1000
weighted avg       0.61      0.57      0.54      1000
```

Figure 4.18: Results.

The neural network obtained lower scores. Only one hidden layer is used for this iteration instead of two as it is too complex for this classification problem.

Synthetic Minority Oversampling Technique

```
Before OverSampling, counts of label '1': [4766]
Before OverSampling, counts of label '0': [17967]

After OverSampling, the shape of train_X: (35934, 74)
After OverSampling, the shape of train_y: (35934,)

After OverSampling, counts of label '1': 17967
After OverSampling, counts of label '0': 17967
```

Figure 4.19

When resampling the data in the first try, the majority class (non-defaulters) was under-sampled. Over-sampling consists of either sampling each member of the minority class with replacement, or creating synthetic members by randomly sampling from the feature set. The models below are trained with a data set after SMOTE.

Logistic Regression

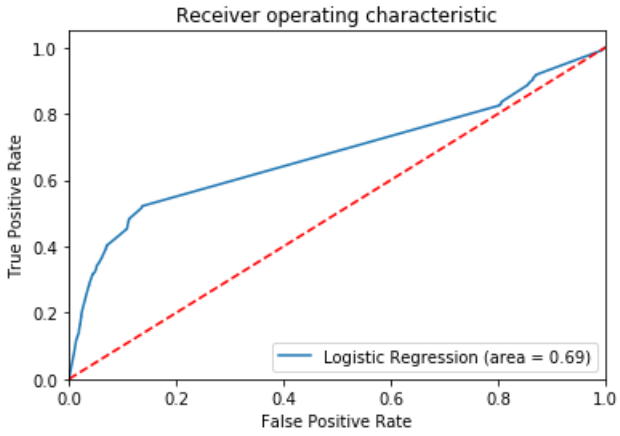
```
=== Confusion Matrix ===
[[4788  757]
 [ 762  813]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.86      0.86      0.86      5545
           1       0.52      0.52      0.52      1575

    accuracy                           0.79      7120
   macro avg       0.69      0.69      0.69      7120
weighted avg       0.79      0.79      0.79      7120
```

(a)



(b)

Figure 4.20: Accuracy increased to 79%.

Neural Network

```
=== Confusion Matrix ===
[[3188 2357]
 [ 470 1105]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.87      0.57      0.69      5545
           1       0.32      0.70      0.44      1575

    accuracy                           0.60      7120
   macro avg       0.60      0.64      0.57      7120
weighted avg       0.75      0.60      0.64      7120
```

Figure 4.21: Results.

Accuracy of neural network increased by a tiny bit to 60%. The model gave a high precision and recall score for 0 and 1 respectively instead, the opposite compared to before SMOTE.

Optimizing the F1 Score

F1 score is optimized instead of a simple average because it creates a balanced classification model with the optimal balance of recall and precision and punishes extreme values [2]. The binary cross entropy loss function of the neural network is replaced with a macro F1 score function from [19] hoping to improve the F1 score of the model.

Neural Network

The neural network is trained on a data set without going through principal component analysis. The theory is that the strength of influence of features will reflect on the weights if a model converges: a feature of weak influence has a weight of smaller value compared to a feature of strong influence.

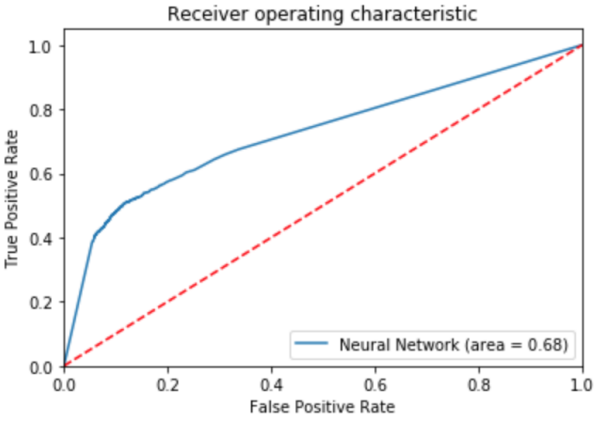
```
=== Confusion Matrix ===
[[5165  380]
 [ 899  676]]


=== Classification Report ===
              precision    recall  f1-score   support

           0       0.85      0.93      0.89      5545
           1       0.64      0.43      0.51      1575

    accuracy                           0.82      7120
   macro avg       0.75      0.68      0.70      7120
weighted avg       0.80      0.82      0.81      7120
```

(a)



(b)

Figure 4.22: Results.

F1 score for both 0 and 1 drastically improved as expected, although the recall score for 1 (default) dropped. Accuracy increased to 82%, the highest of all previous models. This could also be due to the absence of PCA.

Chapter 5

Discussion

5.1 Performance Evaluation

Table 5.1: Performance comparison.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	AUC (%)
Logistic Regression	79	79	79	79	69
Random Forest	65	74	65	61	77.9
Neural Network	82	80	82	81	68

Weighted average scores of each model in the second testing iteration are compared in Table 5.1. The table indicates that random forest is the best model. This structure of trees is adapted to classification problems, and is sophisticated compare to logistic regression, as statistical and machine learning techniques such as bootstrap is involved. The other models are fairly successful too - with AUCs close to 70% - in predicting if a client will default in the following month.

The only unforeseen aspect is the fact that the neural network model performed better with a data set without the dimensionality reduction of principal component analysis. It is to be expected because links representing the features that don't contribute to the network become zero or close to zero. A zero weight means the signals don't pass, so the link is effectively broken. Therefore it is theoretically unnecessary to reduce the dimensions/features for a neural network.

5.2 Evaluation Metric

Model Evaluation Metric

Classification accuracy is determined by calculating percentage number of people defaulted in each class (low, medium, high) in 10 months.

Business Evaluation Metric

- Compare our model score of customers to other credit agencies/services to measure their similarity, current credit service adopted is CTOS.
- Test our model practically by offering devices to low risk customers and study the default rate.

5.3 Challenges

There were several challenges which contributed to the complexity of the problem. One of the biggest obstacles was the relatively limited size of the data set, in particular number of observations of defaulters as mentioned in 4.4. This was a major cause for the poor performance in predicting defaulters, as the models did not have enough data to properly "learn" how to identify a defaulter.

The random forest model is also absent in the SMOTE section, due to the lack of computational power, as a random forest model takes the longest time to train compare to the two other models - up to 7 times the amount of time taken. This is because tons of decision trees have to be constructed for the model to work. This could change the outcome of the performance results. If random forest was trained on a data set that went through SMOTE, it might give an AUC score lower than the other two models, possibly due to overfitting. Neural network seemed to be a more promising model (the highest recall score out of all), as it is a 3 dimensional model whereas logistic regression and random forest are 2 dimensional.

5.4 Future Work

There were some ideas for improvement to go on with this project. As mentioned in Section 3.10, it could be interesting to try to add new variables associated with the building blocks of credit risk, for example the balance-to-limit ratio, and see if it improves the predictive performances of the models. We can also set a precision/recall target score that aligns with our objective - high recall score to predict as many defaulters as possible, and train the model towards the target by optimizing the loss functions and/or weights. Besides, according to literature, gradient boosting handle heterogeneous data well and offer good performance for credit scoring problems. Comparing its predictive power with the models above could put our results into more perspective.

As seen on Table 2.1, machine learning models could be trained to create credit scores by using thousands of data variables from customer information such as social media posts, geolocations, browsing activities, and other data points [20]. Some ideas have been experimented, including using geospatial modelling to predict client income through property value, using mobility data to determine a client's way of transport and using natural language processing to learn a client's browsing activity, and showed promising results.

Chapter 6

Conclusion

A model with a fairly accurate predictive power is implemented. With an AUC of 0.78, random forest beats the other models used. Huge companies out there like Citigroup, Starbucks and PayPal are already benefiting after implementing AI solutions to their problems. The best approach to develop a top-notch classifier is through imagination, creativity and innovation, as there are infinite number of variables and combinations in the real world that may be an indicator to a classification problem. For example, searching some specific words online or the battery health status of a person's phone are found to have influence on a client's creditworthiness! Creditworthiness of a client is dynamic and ever changing. Therefore, temporal data has to be utilized on top of the static data in order to see the bigger picture and into the future.

Bibliography

[1] Shruti Saxena - Precision vs Recall, 2018.
[2] Will Koehrsen - Beyond Accuracy: Precision and Recall, 2018.
[3] Classification with Neural Networks: Is it the Right Choice? - https://missinglink.ai/guides/neural-network-concepts/classification-neural-networks-neural-network-right-choice/
[4] Jason Brownlee - A Gentle Introduction to Cross-Entropy for Machine Learning, 2019.
[5] Multiple Hypothesis Testing and False Discovery Rate - https://www.stat.berkeley.edu/ hhuang/STAT141/Lecture-FDR.pdf
[6] NCSS Statistical Software - Chapter 546, One ROC Curve and Cutoff Analysis.
[7] Prasad Patil - What is Exploratory Data Analysis?, 2018.
[8] Default of Credit Card Clients Dataset - https://www.kaggle.com/netzone/default-credit-and-credit-score-prediction/report
[9] Rajesh S. Brid - Logistic Regression, 2018.
[10] Jason Brownlee - Logistic Regression for Machine Learning, 2016.
[11] Logistic regression - https://www.medcalc.org/manual/logistic_regression.php
[12] Logistic Regression - https://christophm.github.io/interpretable-ml-book/logistic.html
[13] Jake VanderPlas - Python Data Science Handbook, 2017.
[14] Random Forest Classifier – Machine Learning - https://www.globalsoftwaresupport.com/random-forest-classifier/
[15] Jake Huneycutt - Implementing a Random Forest Classification Model in Python, 2018.
[16] Brendan Tierney - Understanding, Building and Using Neural Network Machine Leaning Models using Oracle 18c.
[17] Parul Pandey - Demystifying Neural Networks: A Mathematical Approach (Part 1), 2018.
[18] Jocelyn D'Souza - Let's learn about AUC ROC Curve!, 2018.
[19] Best loss function for F1-score metric - https://www.kaggle.com/rejpalcz/best-loss-function-for-f1-score-metric/comments
[20] Raghav Bharadwaj - Automated Loan Processing and Underwriting – What's Possible Today, 2019.