Imbalanced Classification: A Machine Learning Approach to Predict Loan Defaults

Vivek Majithia

September 2020

School of Mathematics,
Cardiff University

A dissertation submitted in partial fulfilment of the requirements for MSc (in Data Science and Analytics) by taught programme.

CANDIDATE'S ID NUMBER	**C1504584**
CANDIDATE'S SURNAME	**Please circle as appropriate** **Mr / Miss / Ms/ Mrs / Rev / Dr / Other …Majithia……**
CANDIDATE'S FULL FORENAMES	**Vivek Majithia**

DECLARATION

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed …Vivek Majithia…….. (candidate) Date …11/09/2020……

STATEMENT 1

This dissertation is being submitted in partial fulfilment of the requirements for the degree of …MSc……(insert MA, MSc, MBA, etc, as appropriate)

Signed …Vivek Majithia…….. (candidate) Date …11/09/2020……

STATEMENT 2

This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A Bibliography is appended.

Signed …Vivek Majithia…….. (candidate) Date …11/09/2020……

STATEMENT 3 –

I hereby give consent for my dissertation, if accepted, to be available for photocopying and for public viewing, and for the title and summary to be made available to outside organisations.

Signed …Vivek Majithia…….. (candidate) Date …11/09/2020……..

STATEMENT 4 - BAR ON ACCESS APPROVED

I hereby give consent for my dissertation, if accepted, to be available for photocopying and for public viewing **after expiry of a bar on access approved by the Graduate Development Committee.**

Signed …Vivek Majithia…….. (candidate) Date …11/09/2020……..

Acknowledgements

I would like to thank my project supervisor Yuhua Li, who has provided fantastic guidance throughout this process. His insights and advice have been instrumental in bringing this project to fruition.

List of Acronyms

PCA	Principal Component Analysis
ROS	Random Oversampling
SMOTE	Synthetic Minority Oversampling Technique
BLSMOTE	Borderline Synthetic Minority Oversampling Technique
SVMSMOTE	Support Vector Machine Synthetic Minority Oversampling Technique
ADASYN	Adaptive Synthetic Sampling
TL	Tomek Links
ENN	Edited Nearest Neighbours
RENN	Repeated Edited Nearest Neighbours
OSS	One-Sided Selection
NCR	Neighbourhood Cleaning Rule
CNN	Condensed Nearest Neighbour
SMOTEENN	SMOTE + ENN from above
SMOTETOMEK	SMOTE + TL from above
ROC-AUC	Area Under the Receiver Operating Curve
LS-SVM	Least-Squares Support Vector Machine
SVM / SVC	Support Vector Machines / Support Vector Classifier
LOF	Local Outlier Factor
KDE	Kernel Density Estimation
BSS	Brier Skill Score
NL	Nonlinear
LR	Logistic Regression
GNB	Gaussian Naive Bayes
LDA	Linear Discriminant Analysis
DT	Decision Tree
NL SVC	Nonlinear Support Vector Classifier
IG	Information Gain
RBF	Radial Basis Function

KNN	K Nearest Neighbours
ET	Extra Trees
RF	Random Forest
GBM	Gradient Boosting Machine
CW	Class Weights
LOOCV	Leave One Out Cross-Validation

Table of Figures

Executive Summary

It was outlined that lending to borrowers with higher chances of default had severe consequences. A significant example of this was The Global Financial Crisis (2008). Loans that have defaulted are called bad, and those that performed well are called good. Such situations could be tackled through the use of credit risk modelling techniques whereby a credit scorecard could be built and the loan applicants credit score assessed to see whether they are suitable for a loan.

Modern applications in industry consisted of framing this problem as a machine learning classification challenge. Specifically, an imbalanced one as there are situations in which there is a bias distribution or an uneven number of examples in each class, i.e. more examples of good borrowers and fewer examples of bad borrowers.

When exploring the data, it was found that the KDE distribution of each numerical variable per class label followed a very similar distribution, which meant that it was necessary to obtain a robust model to differentiate between the classes. This was solidified through a scatter matrix where the green dots represented the good clients, and the red dots represent the bad clients. It was not easy to see a clear class separation between the green and red dots, thus further cementing a need for a robust algorithm to distinguish between the classes.

Since some variables appeared to have Gaussian-like conveyances, others with apparently exponential or discrete appropriations, standardising or normalising the distributions was valuable for specific machine learning algorithms and the utilisation of the Yeo-Johnson power transformations to make the distributions appear more Gaussian. All models were subject to a mix of data transforms in order to see what brought about the best performance. Due to a variety of features present, dimensionality reduction was also performed to capture 95% of the variance, i.e. PCA.

The data split method of choice was the repeated stratified k-fold cross-validation method, i.e. ten folds with three repeats. More importantly, each data preparation method for all machine learning models highlighted (including sampling methods, PCA, power transformations and score evaluation) occurred within each fold through a pipeline to prevent data leakage.

If probabilities were to be predicted, the use of the Brier Skill Score was highlighted as an efficient metric of measurement as it focused on the positive class. It was determined as the mean squared error between the normal probabilities for the positive class and the anticipated probabilities.

If class labels were the preferred output, the F2 metric was the ideal choice for this project as it was riskier to give a loan to a bad client who appeared to be good than to an actual good client who was mistakenly classified as bad. Thus, the focus needed to be on reducing the number of false negatives. The mean score across the ten folds with three repeats was produced along with a standard deviation for each machine learning model.

Various methods were taken to tackle the class imbalance using standard machine learning algorithms and imbalanced machine learning algorithms such as oversampling using ROS, SMOTE, BLSMOTE, SVMSMOTE and ADASYN. Moreover, undersampling approaches such as TL, ENN, RENN, OSS and NCR and combined approaches such as SMOTEENN and SMOTETOMEK were performed. Cost-sensitive learning and one-class algorithms were also tested and compared against a dummy classifier.

It was found that the standardised calibrated Platt scaled extra trees with balanced class weights performed the best from its range if predicting probabilities was the focus. However, the normalised logistic regression model with a SMOTEENN sampling strategy performed the best from its range if predicting class labels were the focus. These were good results if compared against the respective dummy classifiers. However, they could have been improved given more computational resources using the relevant future works mentioned in Chapter 8.

Table of Contents

Chapter 1: Introduction and Background

Lending to borrowers with a high probability of default was one of the main reasons for severe financial crises such as The Global Financial Crisis (2008). The likelihood that a borrower would not repay their loan to the lender is called “Credit Risk”. In this case, the lender would not receive the owed principal. Moreover, they would not be paid the interest due to and will suffer a substantial loss.

The main factor for the 2008 crisis was high default rates of subprime home mortgages in the United States. First, low-interest rates encouraged financial institutions (lenders) to increase mortgage lending. Banks were willing to finance a hundred per cent or more of the value of a new home. High mortgage approval rates increased the demand for homes which led to an increase in housing prices. Because the value of homes had increased, many homeowners borrowed money and used their homes as a guarantee to the bank (Amadeo 2020).

However, a lot of them were with relatively high credit risk subprime and at some point could not continue repaying their debt and defaulted. Consequently, financial instruments based on mortgages such as mortgage-backed securities lost value, i.e. the residence itself does not “cover” the debt. Thus big banks holding these instruments absorbed huge losses. Some of them like Lehman Brothers and Bear Stearns went bankrupt while the government bailed out many others. That is why credit risk itself is one of the most critical variables in the financial system of today. It is usual for lenders to incur credit losses from every portfolio or exposure over a given period. Such expected loss comes at a result of borrower-specific factors and the economic environment.

Besides, lenders may also incur unexpected losses which are most likely as a result of adverse economic circumstances. It is highly unlikely but still possible that lenders suffer exceptional (stress) losses as a result of a severe economic downturn. Estimating the expected loss (the amount a lender might lose by lending to a borrower) is what is associated with credit risk.

Typically, loans that have defaulted are called bad, and those that performed well are called good. Default definitions are based on the delinquency of the borrower measured in days past the payment due date (commonly ninety days past due date). However, this is not set in stone. A borrower could be defined as a defaulting borrower if they committed fraud too.

Classification is a common task within predictive modelling and involves predicting a class label for a specific example. However, there are situations in which there is a bias distribution or an uneven number of examples in each class. This is known as imbalanced classification, and it poses a challenge to most machine learning tasks.

This is because most machine learning algorithms are tailored towards dealing with similar class distributions. Applying them in an imbalanced classification scenario may cause poor performance on the minority class and have a bias towards the majority class. For specific imbalanced classification challenges, the minority class is given a higher priority than the majority class.

For this project, a standard imbalanced dataset alluded to as the German Credit dataset is used. The dataset was utilized as a significant aspect of the Statlog venture, a European-based activity during the 1990s to assess and think about a considerable number of machine learning models on a scope of various classification errands.

In the case of this project, it is riskier to give a loan to a bad client (minority class) who appears to be good than to an actual good client (majority class) who is mistakenly classified as bad. This requires cautious determination of an evaluation metric that both advances limiting misclassification errors by and large, and favours limiting one sort of misclassification error over another. Hence it is imperative to know whether a client is likely to default on a loan or not. The advancements in machine learning have led to the developments of techniques that can handle the class imbalance issue. This dissertation aims to outline the various techniques used to tackle

the loan default issue using supervised and unsupervised machine learning approaches and their advancements through the use of relevant literature.

Chapter 2: Literature Review

According to (He et al. 2013), imbalanced classification brings about new difficulties and raises various questions within 'real world' applications of predictive modelling. Many of these tasks can be found being tackled within various industries ranging from financial, security, defence, biomedical and more.

It was found by (Olson 2005) in his paper where three experiments were performed on imbalanced data that general algorithms tend to have a bias in their predictions which results as the most common outcome, i.e. majority class being predicted when the class imbalance issue was not tackled. Secondly, it was found that poor performance was bound to be seen in some instances when undersampling was performed as it included the removal of important information. Furthermore, the accuracy rate declined when the data was balanced. The author concluded that decision trees were a stable algorithm when balancing strategies were to be applied.

In addition to this, there was further work done on eight credit scoring data sets by (Baesens et al. 2003). The authors utilised various classification algorithms such as neural networks, logistic regression, k-nearest neighbours, discriminant analysis and more. In this paper, the performance was assessed using accuracy and area under the receiver operating curve (ROC-AUC) as evaluation metrics. It was found that an adjustment to the support vector machine algorithm known as least-squares support vector machines (LS-SVM) and neural networks worked the best. However, simpler classifiers such as logistic regression and linear discriminant analysis also worked quite well.

On the contrary, works using neural networks, linear discriminant analysis, genetic algorithms and decision trees done by (Yobas et al. 2000) concluded that linear discriminant analysis

outperformed the other three models. Whereas, (Desai et al. 1996) mentioned in their paper that linear discriminant analysis was not as effective as using neural networks.

Various sampling strategies have been discussed within relevant literature to address the class imbalance issue. One such example is the synthetic minority oversampling technique (SMOTE) proposed by (Chawla et al. 2002) utilised on a fraud data set as well as detecting oil spills from satellite pictures. The authors state that this method of oversampling the majority class and undersampling the minority class performs better than simply undersampling the majority class.

Moving on, other oversampling strategies include random oversampling as highlighted by (Branco et al. 2016). The technique known as borderline synthetic minority oversampling technique (BLSMOTE) as adapted by (Han et al. 2005) was found to have a better true positive rate and F-score compared to the traditional SMOTE strategy.

Borderline SMOTE-SVM (SVM SMOTE) as represented by (Nguyen et al. 2011) was found to perform better than most oversampling strategies. Adaptive synthetic sampling (ADASYN), which was spoken of by (He et al. 2008) was tested over five evaluation metrics. The authors found that this method decreased the bias towards the majority class examples and that the decision boundary was more focused towards the difficult class examples.

In regards to undersampling techniques, the Tomek links removal approach was studied and suggested by (Tomek 1976) which effectively concluded that the approach resulted in the removal of specific pairs of examples to form a better decision boundary between the two classes.

Edited nearest neighbours (ENN) was an approach highlighted by (Wilson 1972). The principle found was to effectively delete observations of the majority class that were bordered by the minority class using the nearest neighbours approach.

Similarly, repeated edited nearest neighbours (RENN) was another technique mentioned by (More 2016). However, it was found in the research that SMOTE and ENN combined with a logistic regression model, and 'BalanceCascade' yielded the best results. The technique of one-sided selection was emphasised by (Kubat and Matwin 2000). Moreover, the neighbourhood cleaning rule (NCR) was highlighted by (Laurikkala 2002). This method was found to be efficient when dealing with difficult minority classes.

An experiment on thirteen different data sets was undertaken by (Batista et al. 2004), which involved various undersampling and oversampling techniques. The findings suggested that oversampling was generally the preferred method of choice and was more accurate than undersampling procedures. In addition to this, it was found that a combination of oversampling and undersampling strategies tend to work quite well too, namely SMOTE with Tomek links removal or SMOTE along with the ENN approach.

In regards to sampling strategies, it is known to be a common challenge within the field of machine learning to know what would be classed as 'sufficient' sample size. Works done by (Weiss and Provost 2003) prove that different classifiers react differently to different class distributions. What was proposed in the paper was a budget-sensitive sampling method which yielded good ROC-AUC results.

Further advancements within the field are spoken of by (Thai-Nghe et al. 2010) who mentioned the use of cost-sensitive learning which is a relatively new concept. In their paper, the authors tested a mix of resampling techniques with a cost-sensitive SVM model. They also tested adjusting the class weights for the cost-sensitive SVM on eighteen imbalanced data sets and found that adjusting the class weights brought about improvements in model performance.

One could also view the problem of imbalanced classification as an anomaly detection method instead of a supervised learning classification problem. (Schölkopf et al. 2001) highlights the use

of an extension to the support vector machines in the case where labelled data is not present. The authors proposed the one-class support vector machine for this case.

Similarly, (Liu et al. 2012) proposed the use of the isolation forest algorithm, which was founded to be faster and less complex. The authors also found that this algorithm performed better than the one-class support vector machines, local outlier factor (LOF) and random forests in terms of ROC-AUC and processing speeds. It was also found that this algorithm worked very well with higher dimensional data.

In addition to this, another algorithm came into play known as minimum covariance determinant. This was mentioned by (Hubert et al. 2018) in their paper, and the algorithm was found to be quite robust and a convenient tool for outlier detection. The authors also describe two extensions to the algorithm, which made it adaptable for higher-dimensional data.

Finally, (Breunig et al. 2000) introduced the local outlier factor and used this technique on real-world data sets. The authors found that this method was ideal and practical in finding meaningful outliers that could not easily be identified with current approaches.

It is important to note that a classifier is just on a par with the measurement used to assess it. On the off chance that an inappropriate measurement to assess model performance is selected, the higher the chances of the model being helpless. Alternatively, in the most pessimistic scenario, be deceived about the performance expectation of the model. Picking a proper measurement is testing for the most part in machine learning, yet it is incredibly hard for imbalanced classification issues. Initially, because a large portion of the standard measurements that are broadly utilised accept a fair class appropriation, and because commonly not all classes, and in this way, not all prediction mistakes, are equivalent for imbalanced machine learning tasks.

Indeed, the utilisation of standard metrics for example 'accuracy' in imbalanced areas can prompt imperfect classification models and may deliver deluding ends since these measures are not

sensitive toward skewed class distributions (Branco et al. 2016). Moreover, as a result of this, evaluation measurements that place attention on the minority class are highly sought after. However, this is difficult since it is the minority class which lacks the necessary quantity of examples required to prepare a robust model.

To be able to choose a suitable metric (or metrics) for the task at hand. One needs to ask themselves whether a class label is required or merely a probability, i.e. the probability of default. This is the underlying question. The choice of either depends on what makes more sense to communicate with a client or management. In this project, both scenarios were covered. Although, it was not possible to predict probabilities with the anomaly detection methods covered within the scope of this project.

Chapter 3: Data Evaluation and Preparation

3.1 Data Description

This data set contained financial, and background information related to clients, and the task was to predict whether a client was a good or bad applicant. The supposition that will be that the errand included anticipating whether a client would pay back a loan. The data set contained 1000 rows of information and 20 input features, i.e. 7 numerical and 13 categorical features. There were two result classes, 1 for good clients and 2 for bad clients. Good clients were the default (negative) class, though bad clients are the exemption (positive) class. The data set contained seventy percent good clients and thirty percent bad clients.

A cost matrix was provided with the data set that issued a penalty to every misclassification mistake for the minority class. An expense of five was applied for a false negative (denoting a bad client as good), and a one was doled out for a false positive (denoting a good client as bad). It was clear from this that the main focus of this classification task was that false negatives needed to be reduced. This usually comes at the expense of an increase in false positives. However, as stated before that, in general, it is riskier to give a loan to a bad client who portrays himself to be good. This was considered while choosing an evaluation metric.

3.2 Data Exploration

	Checking	Duration	History	Purpose	Amount	Savings	Employment	Intall_rate	Status	Debtors	...
0	A11	6	A34	A43	1169	A65	A75	4	A93	A101	...
1	A12	48	A32	A43	5951	A61	A73	2	A92	A101	...
2	A14	12	A34	A46	2096	A61	A74	2	A93	A101	...
3	A11	42	A32	A42	7882	A61	A74	2	A93	A103	...
4	A11	24	A33	A40	4870	A61	A73	3	A93	A101	...

Figure 1: Example of rows of data from the German dataset

An initial view of the data set showed that the categorical variables were encoded and so a one-hot encoding of these features was required. Likewise, it was observed that the numerical features had various scales; for example, the column 'Duration' had values on a different scale to the column 'Amount'. Thus, scaling of the numerical features was necessary as specific models are sensitive to the scale of the data. The target feature contained estimations of 1 and 2 which also needed to be label encoded to 0 and 1 respectively, to ensure that the standards for imbalanced binary classification errands were met, and where 0 spoke to the negative case and the value 1 spoke to the positive case.

All machine learning models require numerical input features. Hence, the categorical features needed to be encoded to a numerical form (integers) before a model could be fit and assessed. The distribution of the numerical features is summarised in the subplots below.

Figure 2: Histogram of Numerical Variables in the German dataset

Each subplot corresponded to the specific numerical feature. Various appropriations could be noticed, some with Gaussian-like conveyances, others with apparently exponential or discrete appropriations. To some degree, standardising or normalising the distributions would be valuable for specific machine learning algorithms and maybe the utilisation of some power transformations to make the distributions more Gaussian.

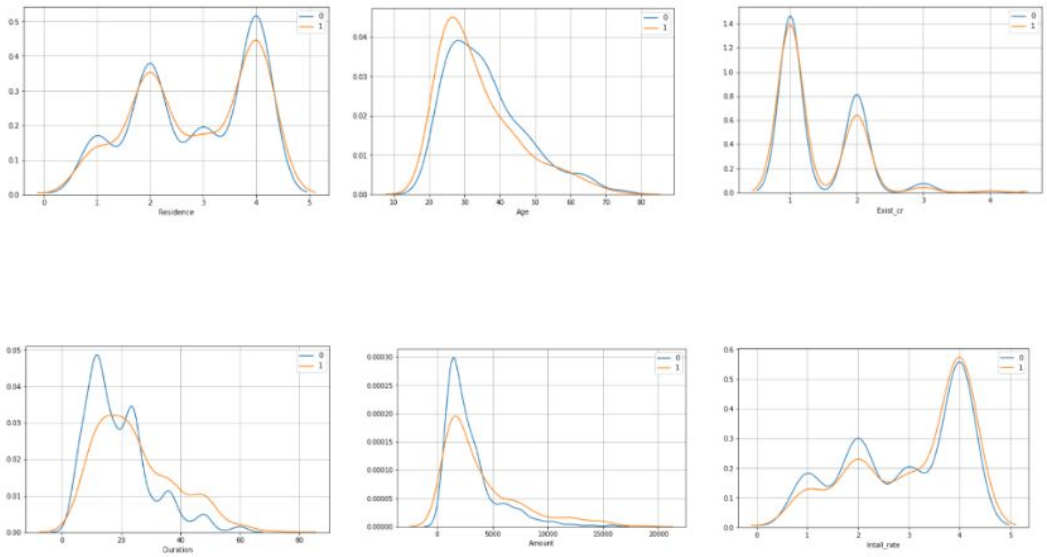

Figure 3: KDE Plots of Numerical Variables by Class

From the six images above, although it was not possible to plot a KDE for the numerical variable 'Maintenance', it was clear that the blue line showed the distribution of the majority class, i.e. 0. and the yellow line indicated the distribution of the minority class, i.e. 1. According to the plots, both classes followed very similar distributions, and it was not easy to see a clear class separation for each variable. Thus, it was imperative that a robust machine learning model was chosen and various techniques applied to be able to tackle this business problem.

To reinforce this, a scatter matrix below also produced the distribution of the numerical variables against each other as well as possible correlations between the variables if any. The green dots represented the good clients, and the red dots represent the bad clients. The density distribution of the features is illustrated across the diagonal. It was not easy to see a clear class separation between the green and red dots, thus further suggesting a need for a robust algorithm to distinguish between the classes. Also noticeable was a positive correlation between the 'Amount' and 'Duration' features for good clients.

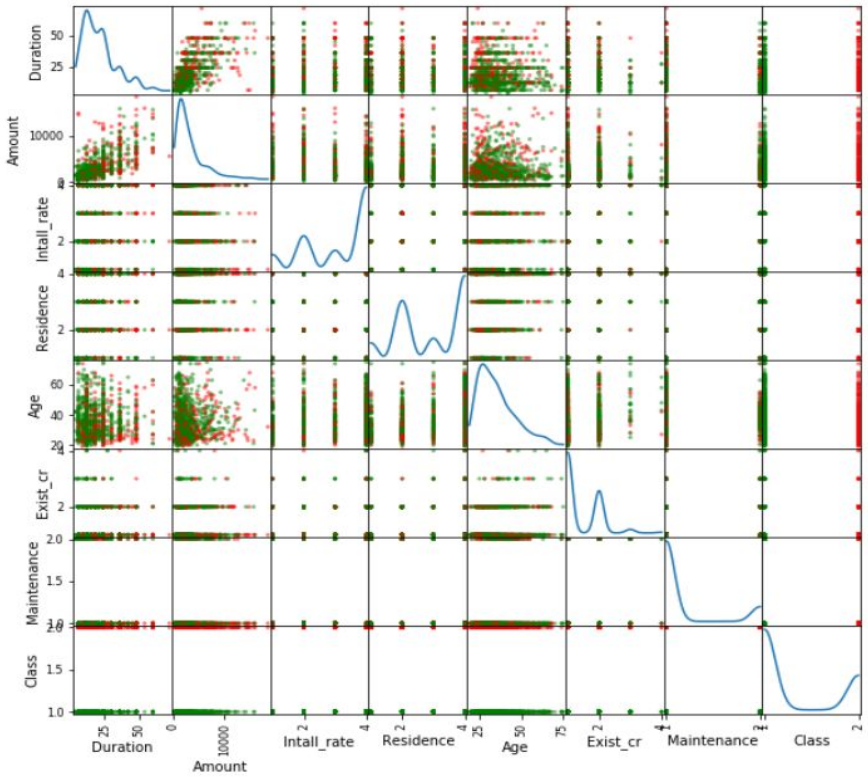

Figure 4: Scatter Matrix of Numerical Variables

Statistical tests for normality were performed to assess whether the variables were normally distributed or not. The tests conducted were the Shapiro-Wilk test, Anderson-Darling test, D'Agostino and Pearson's test. The null hypothesis stated that the feature samples looked Gaussian. The alternate hypothesis stated otherwise. All tests had sufficient statistical evidence to reject the null hypothesis.

Spearman correlations of variables could also be explored through a heatmap. It was noticeable that the 'Amount' and 'Duration' column had a positive correlation. A more accurate representation of visualising correlations is shown below.

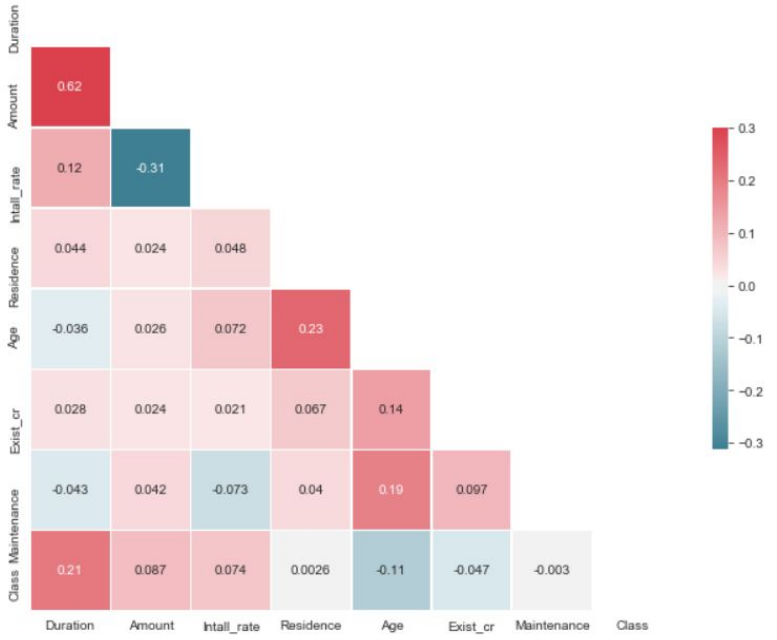

Figure 5: Correlation Matrix of Numerical Variables

It was now clear that 'Amount' and 'Duration' had a positive correlation of 0.62 and some things that were not easily noticeable from the scatter matrix was the weak positive correlation between 'Residence' and 'Age' of 0.23 and the weak negative correlation between 'Amount' and 'Installment Rate' of -0.31.

3.3 Data Preparation

Some machine learning models perform better when features are scaled to a specific range. The two most well-known methods for scaling numerical features before fitting a model are standardisation and normalisation.

Attributes are frequently normalised to lie in a fixed range, for the most part from zero to one by dividing all values by the maximum value experienced or by taking away the minimum value and partitioning by the range between the maximum and minimum values (Witten and Al, 2017). Normalising a value is represented as follows:

$$y = \frac{x - min}{max - min}$$

Another strategy is to compute the statistical mean and standard deviation of the trait values, take away the mean from each value and divide the outcome by the standard deviation. This procedure is called standardising a statistical feature and brings about a set of examples whose mean is zero, and the standard deviation is one (Witten and Al, 2017). Standardising a value is represented as follows:

$$y = \frac{x - mean}{standard\ deviation}$$

Machine learning models take in a planning from input features to a target variable. In that capacity, the scale and distribution of the information drawn from the space might be diverse for every feature. Input features may have various units (e.g. meters, millimetres, and hours) that, thus, may mean the features have various scales. Contrasts in the scales across input features may expand the trouble of the issue being displayed. In practice, it is always a good idea to apply pre-processing transformations to input features before it is presented to a model according to (Bishop, 2002).

A typical obscure inquiry is whether to normalise or standardise a variable or set of variables. However, this project assessed models on data prepared with each transform change and utilised the transform or mix of transforms that brought about the best performance on a model for the given data set.

Many machine learning models perform better when the feature distribution is some-what Gaussian as it is believed that observations for every feature are drawn from a probability distribution. A Gaussian distribution tends to have a natural bell-like shape. In fact, a few models expressly expect the features to have a Gaussian distribution such as linear and logistic regression. Other nonlinear and tree based models might not have this presumption, yet frequently perform better when the features have a Gaussian distribution.

Another regular purpose behind transforming data is to expel distributional skewness and make the data roughly symmetric. Power transforms make the probability distribution of a feature progressively Gaussian. This is regularly portrayed as expelling a skew in the distribution or more formally known as stabilizing the distribution variance. Power transforms can be applied directly by figuring the log or square root of the feature, despite the fact that this could not be the best transform for a given variable (Kuhn and Johnson, 2013).

Rather, one could utilize a summed up form of the transform that finds a parameter which best changes a variable to a Gaussian like probability distribution. A famous methodology is known as the Yeo-Johnson transform which was utilised in this project as it did not require the examples for each feature to have been carefully positive. It underpinned zero values and negative values. The changed data set would then be passed on to a machine learning model to tackle a predictive modelling challenge. Different transforms were undertaken during the data preparation for this project such as standardizing, normalizing, normalizing along with a power transform, standardising along with a power transform. All models were subject to the mix of data transforms in order to see what brought about the best performance.

An example of how the Yeo-Johnson power transform works to make a variable Gaussian like is shown on the 'Amount' variable below:

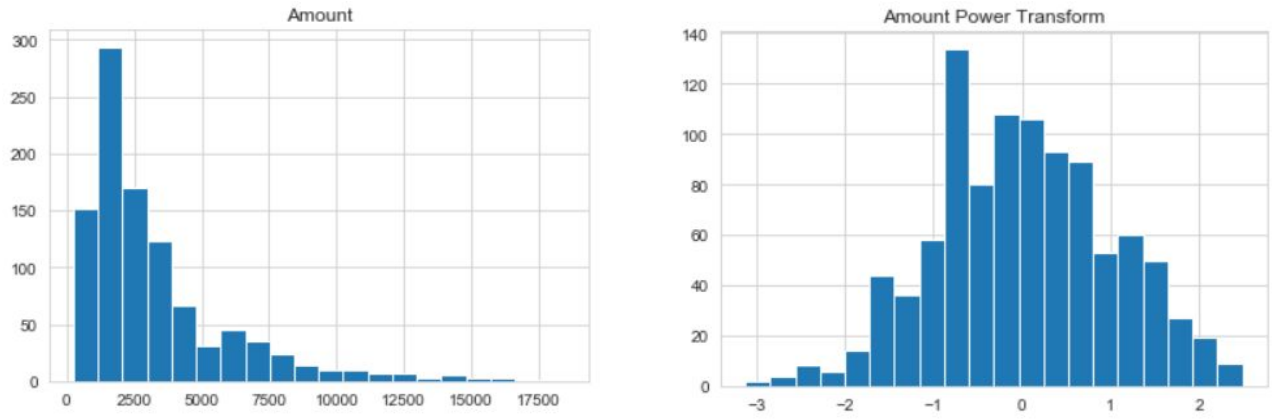

Figure 6: Yeo-Johnson Power Transform Example

3.4 Dimensionality Reduction

Decreasing the quantity of features for a machine learning challenge is alluded to as dimensionality decrease/reduction. Less features can bring about a less complex model that may have better execution when making predictions on new information. Maybe the most mainstream strategy for dimensionality decrease is Principal Component Analysis (PCA) (Murphy and Massachusetts Institute Of Technology, 2012).

PCA can be characterized as the symmetrical projection of the information onto a lower dimensional straight space which is known as the principal subspace, with the end goal that the variance of the projected information is amplified (Bishop, 2013). In essence, it is an unsupervised learning method which is used to identify the interrelations and thus the underlying structure among a set of variables. It is also commonly known as factor analysis. Factor analysis looks for numerous orthogonal/perpendicular lines of best fit to the data set which are perpendicular to each other in n-dimensional space (variable sample space). The most challenging part of PCA is being able to interpret the components.

However, the question is how to choose the right number of dimensions? According to (Aurélien Géron, 2019), it is commonly better and simpler to pick the number of dimensions that sum up to a specific part of the variance for instance 95% rather than testing the number of components to use from the range of the dataset features (this may be a good option if the feature space is relatively small).

Chapter 4: Metric Selection

4.1 Predicting Probabilities

A classification task includes anticipating a class mark for a set of examples. On specific issues, a new class name isn't required, and instead, a likelihood of achieving that class is liked. The likelihood sums up the probability (or vulnerability) of a specific example having a place with each class mark. Probabilities provide more information and can be deciphered by a human administrator or a framework in the industry. Assessing a model dependent on the anticipated probabilities necessitates that the probabilities are adjusted/calibrated. A few classifiers are prepared to utilise a probabilistic system, implying that their probabilities are adjusted. Multiple nonlinear and ensemble classifiers are not prepared under a probabilistic system and consequently require their probabilities to be adjusted against a data set before being assessed by means of a probabilistic measurement (Fernández et al. 2018).

Probabilities are adjusted by re-scaling their qualities, so they better match the distribution seen in the training information. There are two fundamental strategies for scaling predicted probabilities known as Platt scaling and isotonic regression. The former is a more straightforward technique and was created to scale the result from a support vector machine (SVM) to probability estimates. Isotonic regression is an increasingly perplexing weighted least squares regression model. It requires additional training information, although it is likewise increasingly incredible and progressively broad.

The Brier score is a useful metric of choice for predicting probabilities for imbalanced classification tasks as it focuses on the positive class. It is determined as the mean squared error

between the normal probabilities for the positive class and the anticipated probabilities. The score always falls between zero and one, where a score of zero denotes a model that has the perfect skill (Fernández et al. 2018).

A typical practice is to change the score utilizing a reference score, for example, the no skill classifier. This is known as the BSS (Brier Skill Score) and is calculated as follows:

$$BSS = 1 - \frac{Brier\ Score}{Brier\ Score_{ref}}$$

Evaluating the reference score would bring about a BSS of zero. This speaks to a no aptitude prediction. Estimates underneath this will be negative and speak to more terrible than no aptitude. Estimates above zero speak to dexterous expectations with a perfect prediction estimation of one. Thus for the prediction of probabilities, the Brier skill score was selected as a metric of evaluating the classifiers performance.

4.2 Predicting Class Labels

Before moving on to the metric of choice of class labels, it was essential first to understand the various types of prediction results a classification model could output. This could commonly be understood from a confusion matrix, i.e. a matrix that shows ways in which the model may be confused in making predictions about a given class and is represented as follows:

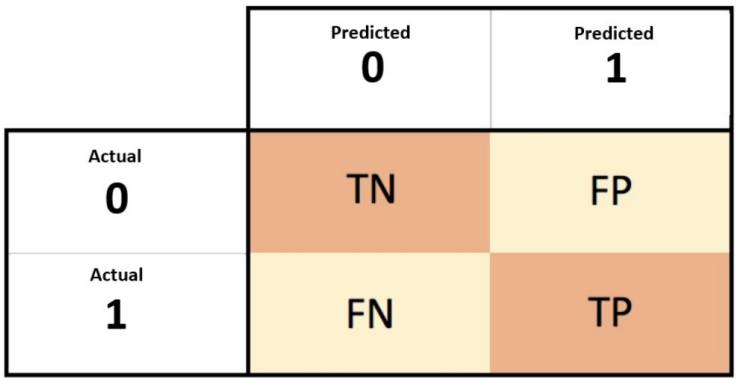

Figure 7: Binary Classification Confusion Matrix (Source)

The actual classes from the data set are given on the left of the matrix and the predicted classes are given on the top of the matrix. The acronyms within the matrix are broken down as follows:

- TN - True Negative
- FN - False Negative
- FP - False Positive
- FN - False Negative

A true negative is where the model correctly predicts the negative class, i.e. classifying a good client as good. Note that negative class means 'no change,' i.e. 0 is usually the negative class, whereas 1 tends to be the positive class as it implies a change has occurred. A false negative is where the model incorrectly predicts the positive class, i.e. classifying a bad client as good. A false positive is where the model incorrectly predicts the negative class, i.e. classifying a good client as bad. Finally, a true positive is where a model correctly predicts the positive class, i.e. classifying a bad client as bad. Earlier it was stated that it was riskier to give a loan to a bad client who appears to be good than to an actual good client who was mistakenly classified as bad. From this statement, it was clear that the focus needed to be on reducing the number of false negatives.

Furthermore, an excellent metric to use when the focus is on reducing false negatives is the F-measure. The F-measure is a technique that consolidates both the recall and precision of a model into a solitary measure that catches the two properties. Precision assesses the division of correctly arranged cases among the ones characterized as positive. The recall is a metric that measures the quantity of right positive predictions made out of predictions for the positive class, i.e. out of all the predictions for the positive class, how many of them were correctly classified as positive. It gives a sign of missed positive expectations. They are calculated as follows:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \qquad Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Thus, the F-measure was formed where both precision and recall are combined to form one metric. Individually, neither recall nor precision is appropriate to convey the entire performance of a classifier. This is because we can have a fantastic precision score but with a horrible recall score or vice versa. But the F-measure gives an approach to communicate the two with a solitary score and ranges from zero (poor) to one (perfect) and is calculated as follows:

$$F - measure = \frac{2 * Precision * Recall}{Precision + recall}$$

The F-measure, which loads recall and precision similarly, is the variation regularly utilized when gaining from imbalanced classification tasks (He et al. 2013). Reducing false negatives meant placing more value on recall and a little less on precision. The F-measure offers a choice of three weight preferences to either precision or recall, which is known as the F Beta-measure:

- F1-measure - Balances the weight on both precision and recall
- F0.5- measure - Adds more weight to precision and less to recall
- F2-measure - Adds more weight to recall and less to precision

As such, the F2-measure was chosen as an ideal metric candidate for predicting class labels and is formulated as follows:

$$F2 - measure = \frac{(1 + 2^2) * Precision * Recall}{2^2 * Precision + Recall}$$

Chapter 5: Standard Machine Learning Algorithms

5.1 Data Split

When dealing with predictive modelling tasks, it is usually common for practitioners to split the data into a typical train and test set using a given ratio that is chosen by the practitioner. The reason for this is because it does not make sense to train the model using all the available data and evaluate the same data. The goal is to test how the model performs on unseen data. Hence in practice, a model is trained on train data and then tested on unseen (test) data (Albon, 2018).

There are two common approaches when it comes to splitting data. One approach is to use a train-test split function with a given ratio such as 70% (train) and 30% (test) dependent on the practitioner's preference or the k-fold cross-validation technique. Both approaches are widely used in practice but could prove to be unreliable when dealing with a class imbalance issue. According to (Allibhai, 2018) in his blog, cross-validation is a much better choice as it allows the model to train on several splits and is usually a better indicator of model performance on unseen data.

K-fold cross-validation works by first splitting the data into k-folds. The choice of 'k' depends on the practitioner, but it is found that a value of ten tends to yield reliable results (Raschka and Olson, 2015). Next, the initial 'k-1' folds are used to train the model and the last 'k^{th}' fold is used to test the model. The process is repeated so that each fold is given an opportunity to be a test set. This implies that ten models would be fit and assessed; the performance is calculated to be the mean of these ten runs.

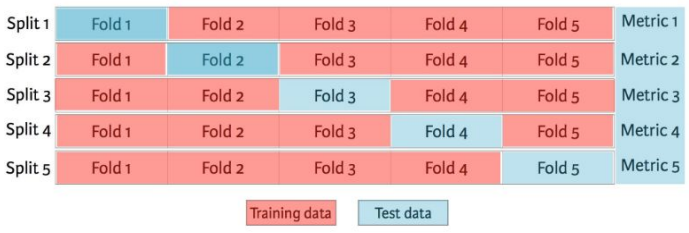

Figure 8: K-fold Cross Validation (Source)

On the contrary, 10-fold cross-validation is inappropriate for class imbalance tasks even if the imbalance is not as extreme. One solution to this would be to use a stratified version of this technique which maintains the appropriate original distribution of positive and negative samples within each of the folds instead of splitting the data randomly (He et al. 2013).

Similarly, an extension to this would be to have the ability to repeat the stratified 10-fold cross-validation technique with randomisation several times for each time. This is known as repeated stratified k-fold cross-validation, and in this project, each stratified cross-validation was repeated with randomisation three times (Venkata, 2018). This simply meant that the 10-fold cross-validation was repeated three times with different randomisation for each repetition to ensure the reliability of the results.

5.2 Linear Algorithms

5.2.1 Logistic Regression Intuition

Adapted from (Albon 2018), a logistic regression model tackles classification tasks that predict a discrete category e.g. binary classification tasks. A linear regression model should not be used on binary classification challenges as it will not lead to a good fit. The linear regression equation is identified as follows:

$$y = b_0 + b_1 x$$

However, the linear regression model can be transformed to a logistic regression model which ranges between zero and one. The sigmoid function takes in any value for 'z' and the result is a value between zero and one. The formula of the sigmoid function is identified as follows:

$$\varphi(z) = \frac{1}{1 + e^{-z}}$$

Thus a linear regression solution can be placed as input for the sigmoid function which results in a probability (p) between zero and one for the positive class.

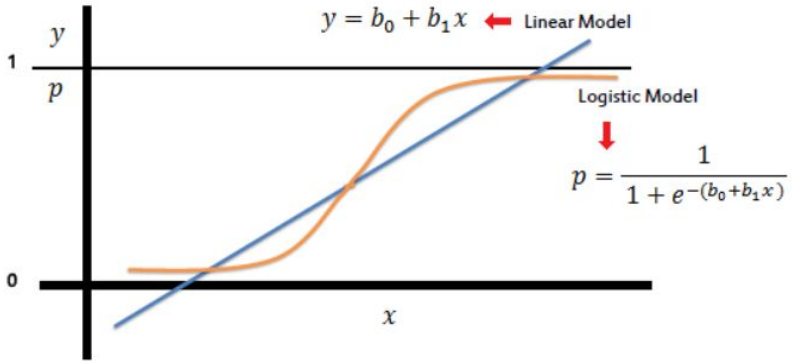

Figure 9: Sigmoid Function with Linear Regression Input (Source)

The default probability threshold is 0.5. This implies that any output value that is less than or equal to 0.5 is classed as zero and any value greater than 0.5 is classed as one.

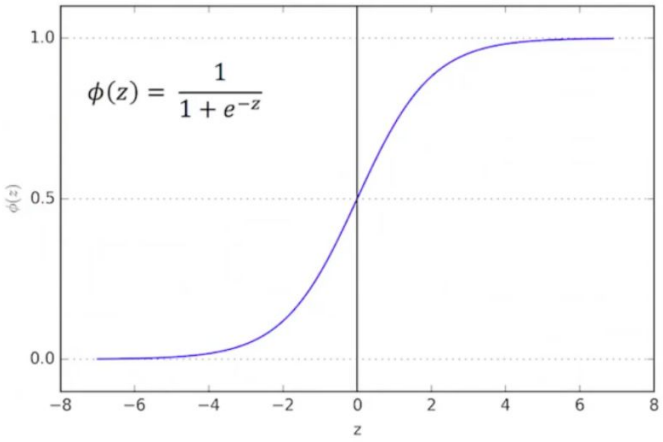

Figure 10: Sigmoid Function (Source)

5.2.2 Gaussian Naive Bayes Intuition

Adapted from (Albon 2018), to understand the probability of a given event knowing previous conditions that could be related to the event, one could turn to the Bayes theorem for answers. It is represented as follows:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

$A,\ B\ =\ Events.$

$P(A \mid B)\ =\ Probability\ of\ A,\ given\ B\ has\ occurred\ or\ is\ true.$

$P(B \mid A)\ =\ Probability\ of\ B,\ given\ A\ has\ occurred\ or\ is\ true.$

$P(A),\ P(B)\ =\ Independent\ probabilities\ of\ A\ and\ B\ respectively.$

One can apply the Bayes theorem twice given a set of features (X). The formulas are represented as follows:

$$P(No\ Default \mid X)\ =\ \frac{P(X \mid No\ Default)\ P(No\ Default)}{P(X)}$$

$$P(Default \mid X)\ =\ \frac{P(X \mid Default)\ P(Default)}{P(X)}$$

$P(No\ Default)\ and\ P(Default)\ =\ Prior\ Probability$

$P(X)\ =\ Marginal\ Likelihood$

$P(X \mid No\ Default)\ and\ P(X \mid Default) =\ Likelihood$

$P(No\ Default \mid X)\ and\ P(Default \mid X)\ =\ Posterior\ Probability$

The first step is to calculate the prior probabilities, which is the number of observations classified as not default and default respectively divided by the total number of observations. Next, the marginal likelihood is calculated as the number of similar observations within the given radius of the data point divided by the total number of observations. To calculate the likelihood, the number of similar observations among those who do not default is divided by the total number of observations. Similarly, the number of similar observations among those who default is divided by the total number of observations. Thus to summarise the equations:

$$P(No\ Default)\ =\ \frac{Number\ of\ Observations\ Classed\ as\ No\ Default}{Total\ Number\ of\ Observations}$$

$$P(Default)\ =\ \frac{Number\ of\ Observations\ Classed\ as\ Default}{Total\ Number\ of\ Observations}$$

$$P(X)\ =\ \frac{Number\ of\ Similar\ Observations}{Total\ Number\ of\ Observations}$$

$$P(X \mid No\ Default)\ =\ \frac{Number\ of\ Similar\ Observations\ Among\ Those\ Not\ Defaulted}{Total\ Number\ of\ Observations}$$

$$P(X \mid Default) = \frac{Number\ of\ Similar\ Observations\ Among\ Default}{Total\ Number\ of\ Observations}$$

The new data point is then compared for each formula of $P(No\ Default \mid X)$ and $P(Default \mid X)$. If the former probability is larger than the latter probability then the data point is classified as a zero. Otherwise it is classed as a one.

5.2.3 Linear Discriminant Analysis Intuition

Adapted from (Raschka and Olson 2015), linear discriminant analysis (LDA) is commonly used as a dimensionality reduction technique and its goal is to project data into a lower dimensional space. Along with calculating the number of components that maximise the variance within the data, LDA also allows for maximising the difference between classes.

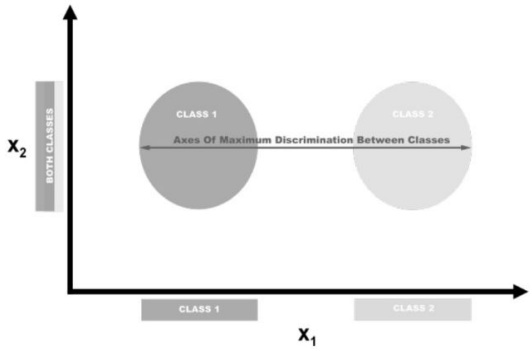

Figure 11: LDA Maximising Component Axes for Class Separation (Source)

From this figure, there are two target output classes and two features for the sake of easy visualisation. It will be noticeable that if the data is projected vertically onto the y-axis then the classes will not have the ease of separability. However, a horizontal projection onto the x-axis preserves the class separability. The concept remains the same as the dimensionality of the data increases.

5.2.4 Linear Support Vector Classifier Intuition

Adapted from (Aurélien Géron 2019), suppose there are points on a two-dimensional plane with some red and blue points. The aim is to be able to derive a line that would be able to separate the

red and blue points as going forward; the decision boundary is critical to decide where a new point would fall. However, there are multiple lines that can be drawn to separate the observations, as shown below:

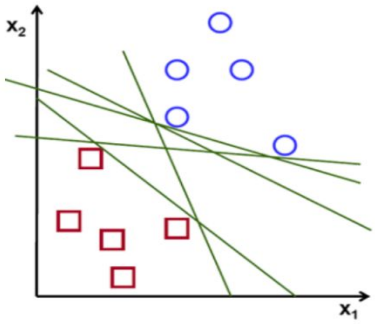

Figure 12: Various Lines Separating the Classes (Source)

However, each one of these lines in the future would have differing consequences. Thus, when new points are added, they would either be classed as blue or red. Thus, the aim is to determine the optimal line to help separate the given space into classes. So how does the support vector machine (SVM) search for this line? The line is searched through the maximum margin as shown below, i.e. the distance between the line and the support vectors are maximised.

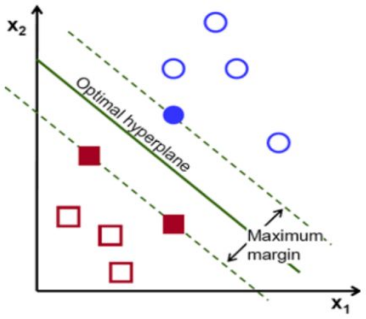

Figure 13: Optimal Hyperplane and Maximum Margin (Source)

The support vectors (points that support the entire algorithm) are the two filled red points and single filled blue points on either side of the line (two-dimensional space) or hyperplane (multidimensional space) and are vectors in a multidimensional space. To the right of the optimal

hyperplane is the positive hyperplane (dashed line) and to the left is the negative hyperplane (dashed line).

Suppose the task at hand would be to classify an orange (blue) from an apple (red). Predominantly machine learning algorithms would try to find the most stock standard common type of apple and oranges; these would be the points furthest away from each other (i.e. the points not filled in). However, an SVM works differently. The algorithm would try to find an apple that looks very much like an orange and vice versa. These form the support vectors and tend to be very close to the decision boundary as well as each other. Thus, an SVM looks at extreme cases close to the boundary and then uses that to construct its analysis. The formula to make a prediction based on a new input using the inner product is:

$$f(x) = A_0 + \sum_{i=1}^{m} (b_i * (x * x_i))$$

$x = New\ input\ vector$

$x_i = Support\ vector$

$A_0\ and\ b_i = New\ input\ coefficients$

The kernel maps the similarity between the support vectors and the new observations. Due to the distance measure being a linear combination of the input variables, the similarity measure used in the case of linear SVM is the inner product and can be re-written as:

$$K(x, x_i) = \sum(x * x_i)$$

5.3 Nonlinear Algorithms

5.3.1 Decision Trees Intuition

Adapted from (Raschka and Olson 2015), if the goal is about interpretability, decision trees are the way forward. It works by decomposing the data, i.e. performing splits and then based on a series of questions; decisions are made. An example of a simple decision tree determining if someone is fit or not is given below:

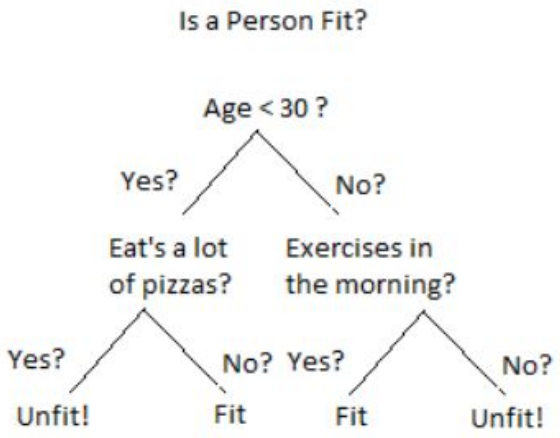

Figure 14: Example of a Decision Tree (Source)

Based on the given variables, the algorithm attempts to learn from a series of questions and perform splits in order to find a distinction between the given classes. This is done at the tree root initially, the model then begins to find splits based on variables that output the largest Information Gain (IG) or merely the split that produces the largest decrease in impurity at a particular decision node. The default measurement of impurity used by decision trees is the Gini impurity which is formulated as follows:

$$G(x) = 1 - \sum_{i=1}^{m} p_i^2$$

$G(x) = Impurity\ at\ node\ x$

$p_i = Proportion\ of\ examples\ of\ class\ m\ at\ the\ node\ x$

The process of splitting is repeated until all the leaves are pure which in turn means that the samples at each of the nodes of the leaf belong to the same class.

5.3.2 Nonlinear Support Vector Classifier Intuition

Adapted from (Aurélien Géron 2019), previously linearly separable classes were mentioned. What happens if the data is not linearly separable? The figure below illustrates a scenario where a linear separation cannot be found:

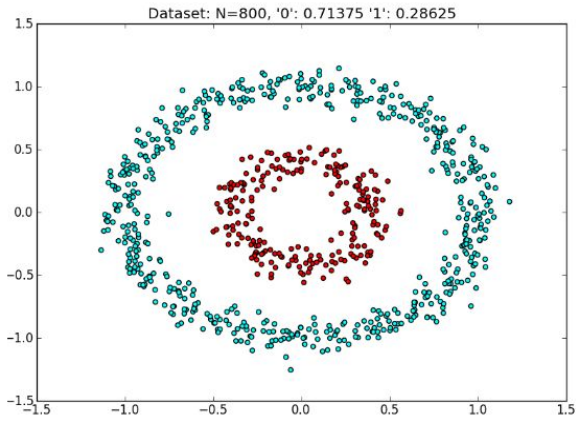

Figure 15: Nonlinear Example (Source)

In a two-dimensional scenario as above, a simple straight line or hyperplane separating the classes cannot be conceived. However, the Gaussian radial basis function (RBF) kernel trick can help create a non-linear separator which is represented as follows:

$$K(x,\ x_i) = e^{-gamma * \sum((x - x_i^2))}$$

Where the gamma parameter which ranges between zero and one is one that needs to be decided as input. A more common value for gamma is 0.1. The result of changing the kernel from linear to RBF for such a scenario would be:

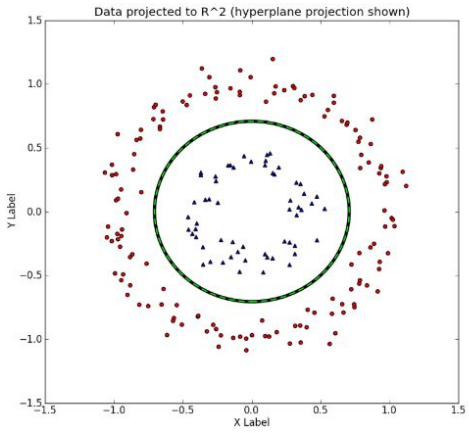

Figure 16: Nonlinear Separator (Source)

5.3.3 K Nearest Neighbours Intuition

Adapted from (Raschka and Olson 2015), suppose for simplicity and illustrative purposes, a two-dimensional dataset has two given target categories and a new data point for prediction, as shown below:

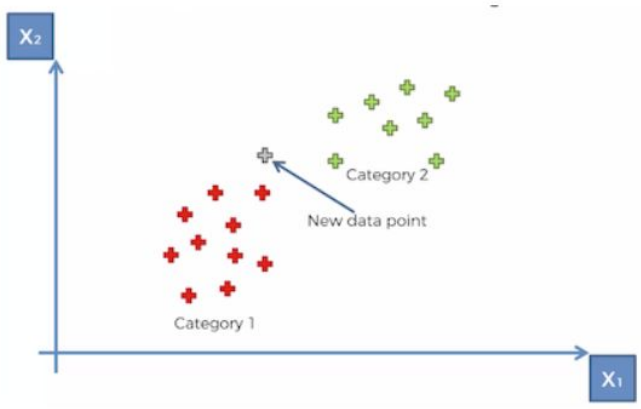

Figure 17: Classifying a New Data Point (Source)

The question then formed is should the new data point belong to the red category or the green one? The steps to determine this using the K Nearest Neighbours approach (KNN) would be to:

- Choose the number of K neighbours (e.g. 3, 5, 7).

- Using Euclidean distance, take the K nearest neighbours of the given data point.
- From the K neighbours, count the number of points from each class/category.
- Assign the new data point to the majority class seen within the given neighbourhood.

The image below illustrates the concept of Euclidean distance:

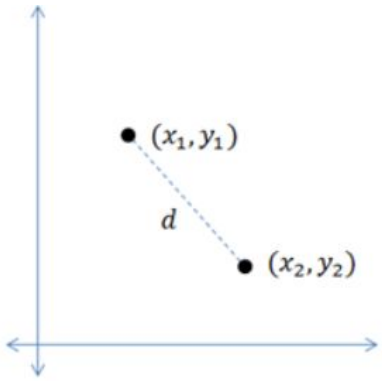

Figure 18: Euclidean Distance (Source)

Given the coordinates (x_1, y_1) and (x_2, y_2), the Euclidean distance between the points is calculated as:

$$Euclidean\ Distance\ (d) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

In higher dimensions, the formula can also be re-written as:

$$d_{L2}(a, b) = \|a - b\| = \sqrt{\sum_{i=0}^{n} (a_i - b_i)^2}$$

5.4 Ensemble Algorithms

5.4.1 Extra Trees Intuition

(Geurts et al. 2006) states that the algorithm is quite similar to the random forest algorithm but simpler at building the decision trees. The algorithm builds various decision trees utilising a random set of features and splits the nodes at random. A vital feature of this algorithm which can be changed based on the practitioner is that there is no replacement with the samples chosen. In other words, it means that the observations are not bootstrapped. It is important to note that the randomness is derived from the random splits, and due to this, decision trees within the algorithm tend to become less correlated with each other. Although, this may increase the variance within the model, which can be reduced by increasing the number of overall decision trees.

5.4.2 Random Forest Intuition

Adapted from (Breiman 2001), random forests create an ensemble of decision trees using bootstrap samples of the training set. Bootstrap samples of the training set essentially mean 'with replacement'. However, when building each tree, each time a split is considered, a random sample of 'm' variables is chosen as a split candidate from the full set of 'p' variables. The split is only allowed to use one of those 'm' variables. Moreover, for each tree at each split, a new random set of variables is selected. In a classification scenario:

$$m = \sqrt{p}$$

Due to the randomness and leaving out certain candidate variables, the decision trees tend to become uncorrelated. Furthermore, due to having many trees, the overall variance of the model is then reduced.

5.4.3 Gradient Boosting Classifier Intuition

A video by (Starmer 2019) states that gradient boosting classification has a lot in common with logistic regression. The classifier begins with a leaf that represents an initial prediction for every

individual which is the log (odds). The log (odds) can also be seen as the logistic regression equivalent of the average. Just like with logistic regression, the easiest way is to convert the log (odds) into a probability which is done using a logistic function:

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

How bad the initial prediction is can be measured by calculating pseudo residuals, which is the difference between observed and predicted values. A tree is then built using a given set of features to predict the residuals. In practice, the number of leaves is set to be between eight and thirty two. In a regression context, a leaf with a single residual has an output value equal to that residual. However, it is a little more complex with classification as the predictions are in terms of log (odds) and the leaves are derived from probabilities. Thus, it is not possible to add them together to get a new log (odds) prediction without some sort of transformation. The most common type of transformation is:

$$\frac{\sum Residual_i}{\sum[Previous\ Probability_i * (1 - Previous\ Probability_i)]}$$

The numerator is the sum of all the residuals in the leaf, the denominator is the sum of the previously predicted probabilities for each residual times one minus the same predicted probability. The predictions are then updated by combining the initial leaf with the new tree which is scaled by a learning rate. The log (odds) prediction is then converted to a probability. Sometimes this probability may be worse than the initial one and that is why lot's of trees are built. New residuals are also then calculated and a new tree with output values for each leaf is then constructed. The process is repeated until the maximum number of trees specified has been met, or the residuals get very small.

Chapter 6: Imbalanced Machine Learning Algorithms

6.1 Oversampling Methods

6.1.1 Random Oversampling

According to (Branco et al. 2016), random oversampling involves duplicating examples of the scarce class, i.e. minority class at random and including them into the data in order to balance the distribution of the target classes for optimum machine learning performance. However, there are chances of overfitting. Thus, it is always a good idea to check the performance on both train and test sets and compare against the baseline performance. The synthetic example below shows the effects of random oversampling on an imbalanced dataset.

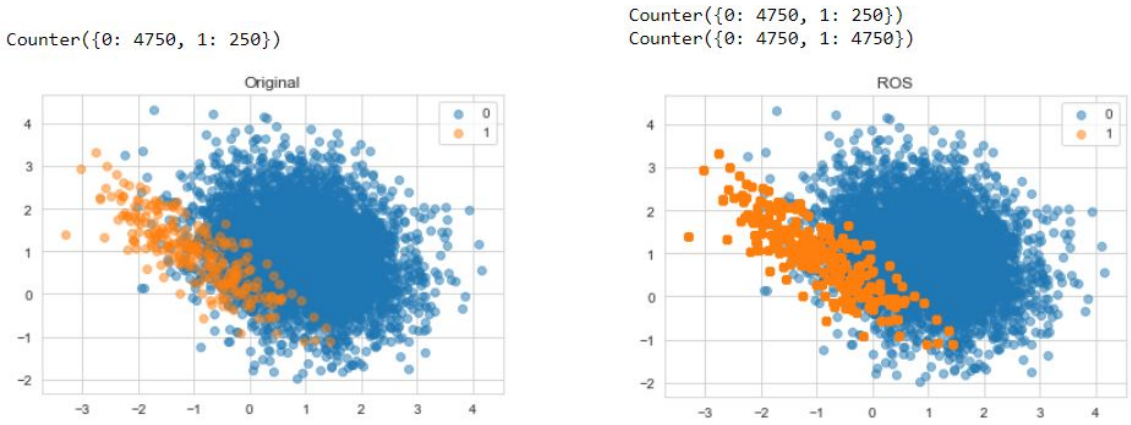

Figure 19: Before and After Random Oversampling

6.1.2 Synthetic Minority Oversampling Technique

Intuitively, this method known as SMOTE utilises a k-nearest neighbours approach. For every example in the minority class, the algorithm finds the nearest neighbours. Based on the number of samples the user wants SMOTE to create, the algorithm will first try to identify the lines which join the minority class samples based on the nearest neighbours. New artificial samples are then created along these lines. After synthesising the new minority instances, the imbalance that initially existed decreases significantly. However, one issue with SMOTE is that the new data points are created without taking into consideration the type of examples within the majority class. Thus these points may become open to more than one interpretation (Chawla et al. 2002). The synthetic example below shows the effect of SMOTE on an imbalanced dataset.

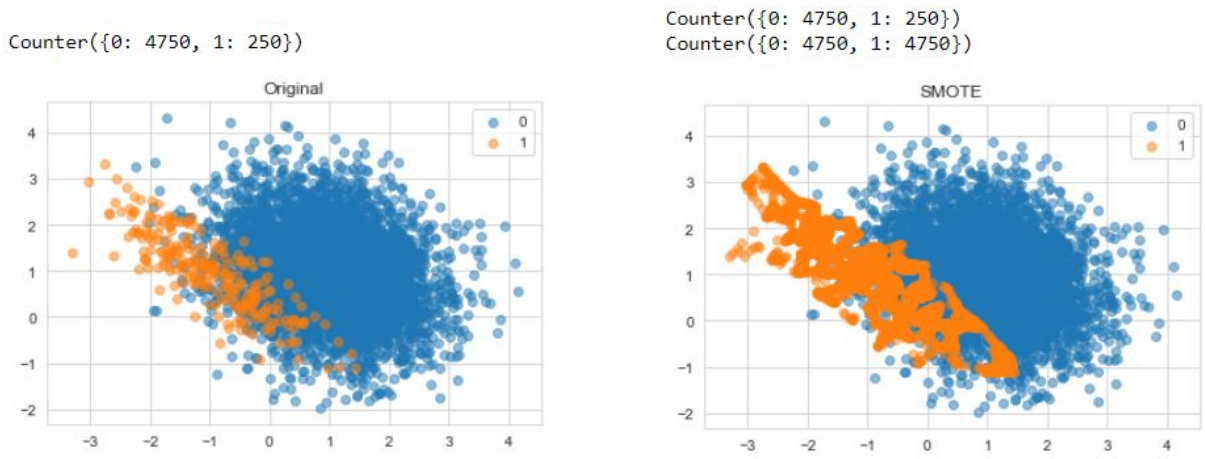

Figure 20: Before and After SMOTE Oversampling

6.1.3 Borderline Synthetic Minority Oversampling Technique

Commonly known as BLSMOTE, which is an adaptation of the SMOTE technique above. BLSMOTE also uses a k-nearest neighbours approach but focuses more towards the examples of the minority class that tend to lie on the boundary where it may be mistaken for the majority class, i.e. the difficult examples. These difficult examples are the focus and are oversampled. Thus, providing more insight into the machine learning algorithm to then differentiate between the classes (Han et al. 2005). The synthetic example below shows the effect of BLSMOTE on an imbalanced dataset.

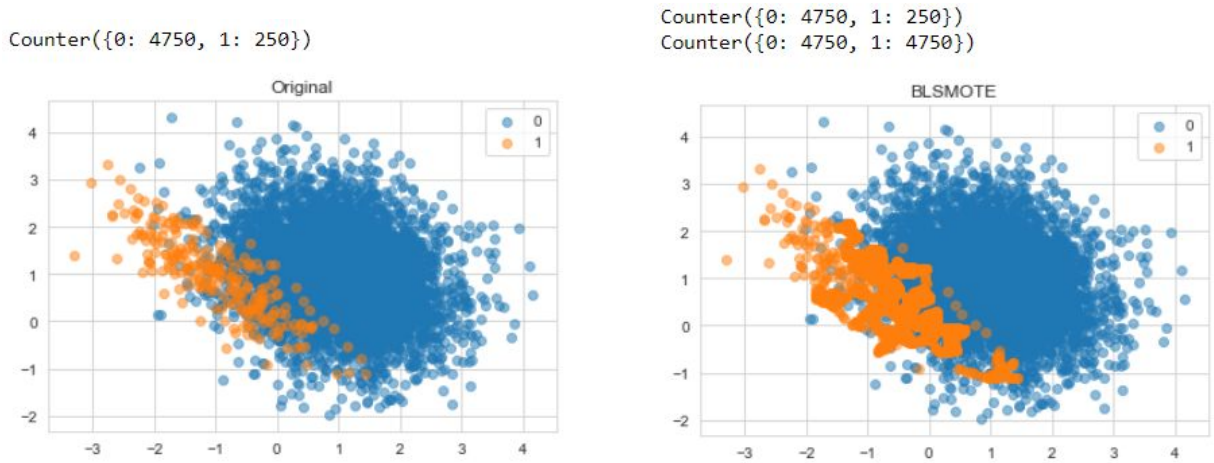

Figure 21: Before and After BLSMOTE Oversampling

6.1.4 Borderline-SMOTE SVM

Also known as SVMSMOTE, which is another adaptation of the traditional SMOTE technique. (Nguyen et al. 2011) mentions that the method follows a similar intuition as the BLSMOTE method. However, a support vector machine algorithm is implemented in finding the decision boundary between the classes instead of the k-nearest neighbours approach. New synthetic examples of the minority class are then created close to the support vectors. The synthetic example below shows the effect of SVMSMOTE on an imbalanced dataset.

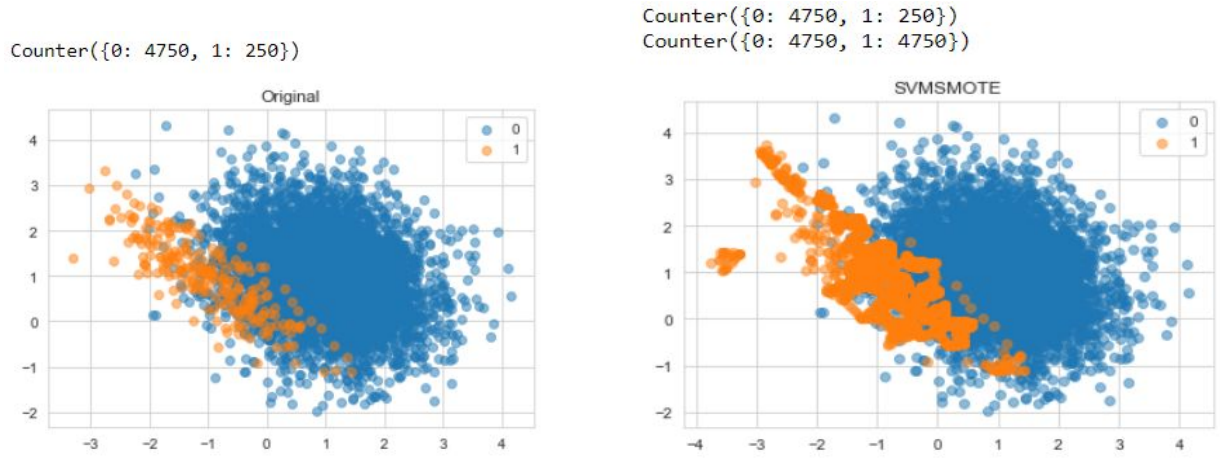

Figure 22: Before and After SVMSMOTE Oversampling

6.1.5 Adaptive Synthetic Sampling

This sampling method known as ADASYN, is a modification to SMOTE. It is relatively similar to SMOTE only that this method generates synthetic samples based on distributions of the minority class. This means that more examples are synthesised in areas where the minority class has a lower distribution, and fewer examples are generated where the distribution of minority examples are more concentrated (He et al. 2008). The synthetic example below shows the effect of ADASYN on an imbalanced dataset.

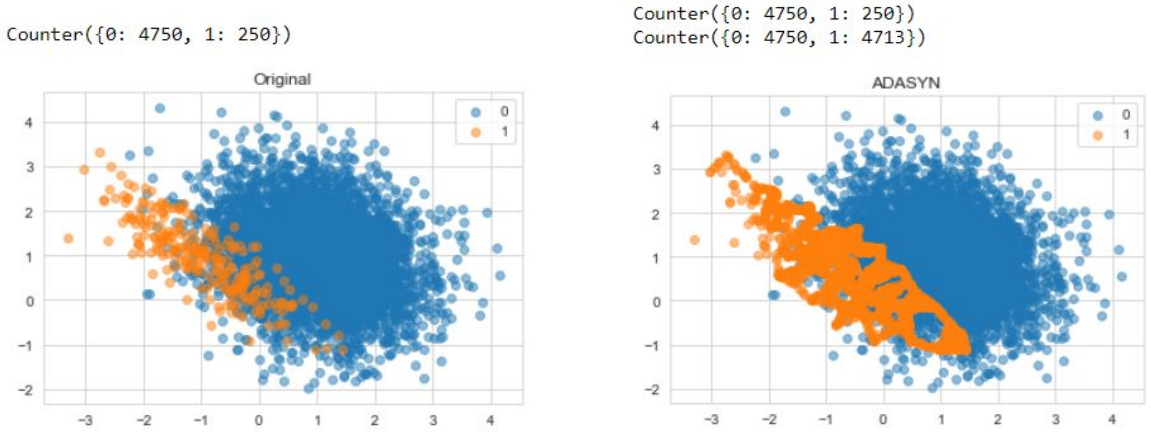

Figure 23: Before and After ADASYN Oversampling

6.2 Undersampling Methods

6.2.1 Tomek Links

Tomek links are pairs of observations from separate classes near the borderline that are relatively close to each other. Removing the pairs allows for a better decision boundary to be formed between the two classes. Interestingly, this method removes both borderline instances as well as unnecessary noise from the data set (Tomek 1976). The synthetic example below shows the effect of Tomek Links on an imbalanced dataset.

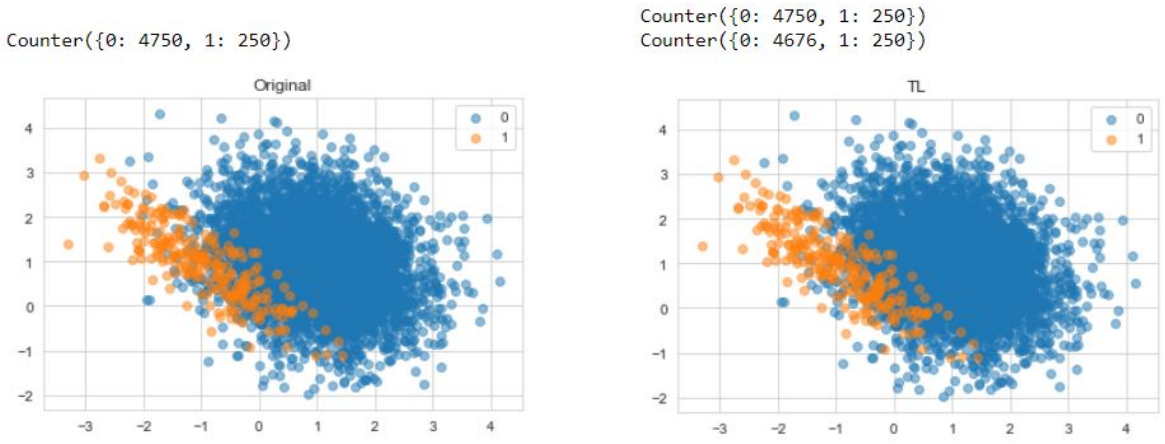

Figure 24: Before and After Tomek Links Undersampling

6.2.2 Edited Nearest Neighbours

ENN as it is known also works using a k-nearest neighbour approach. It works by making an observation and identifying its class. Next, the method then checks the nearest neighbours (usually three neighbours) and checks if at least two of the three neighbours are of a differing class. If this condition is satisfied and the observation is from the majority class, then the observation is deleted. However, if the observation is from the minority class and at least two out of the three neighbours are from the majority class, then those majority class observations are deleted. The principle is to effectively delete observations of the majority class that are bordered by the minority class using the nearest neighbours approach (Wilson 1972). The synthetic example below shows the effect of ENN on an imbalanced data set.

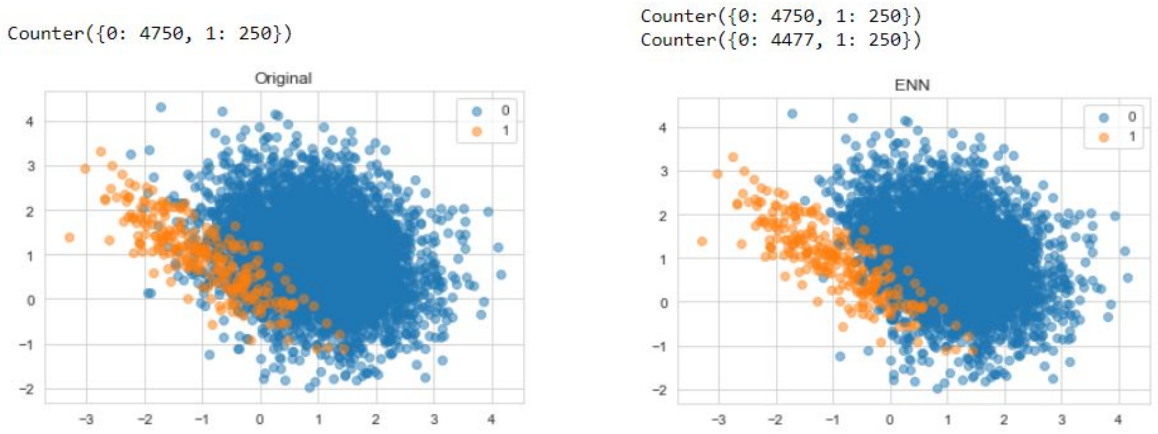

Figure 25: Before and After ENN Undersampling

6.2.3 Repeated Edited Nearest Neighbours

RENN works by applying the ENN algorithm continuously until no more examples can be removed (More 2016). The synthetic example below shows the effect of RENN on an imbalanced dataset.

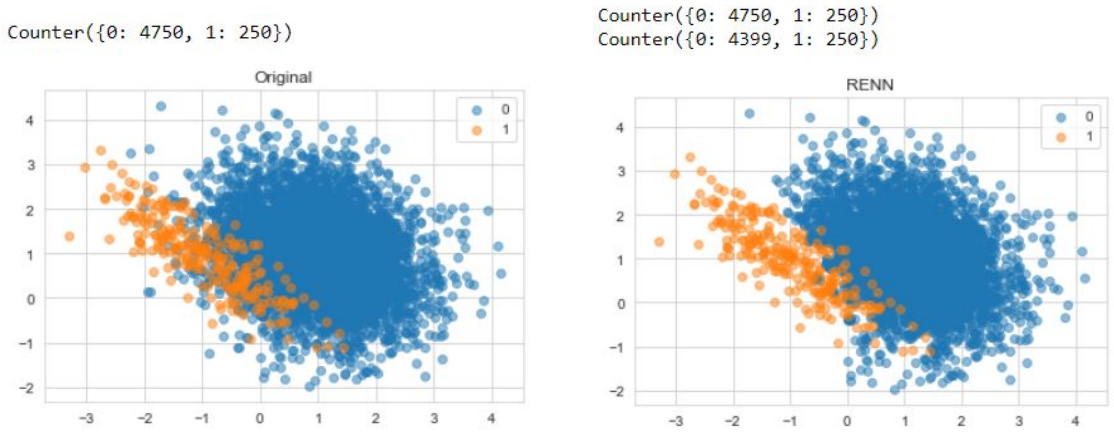

Figure 26: Before and After RENN Undersampling

6.2.4 One-Sided Selection

OSS for short combines both the Tomek links and another approach called the condensed nearest neighbour (CNN). Firstly, the Tomek links remove the obscure pair of observations that allow for a better decision boundary to be formed between the classes. Finally, the unnecessary majority class observations which are far away from the separating boundary are then deleted using the CNN approach as highlighted by (Kubat and Matwin 2000) in their paper. The synthetic example below shows the effect of OSS on an imbalanced dataset.

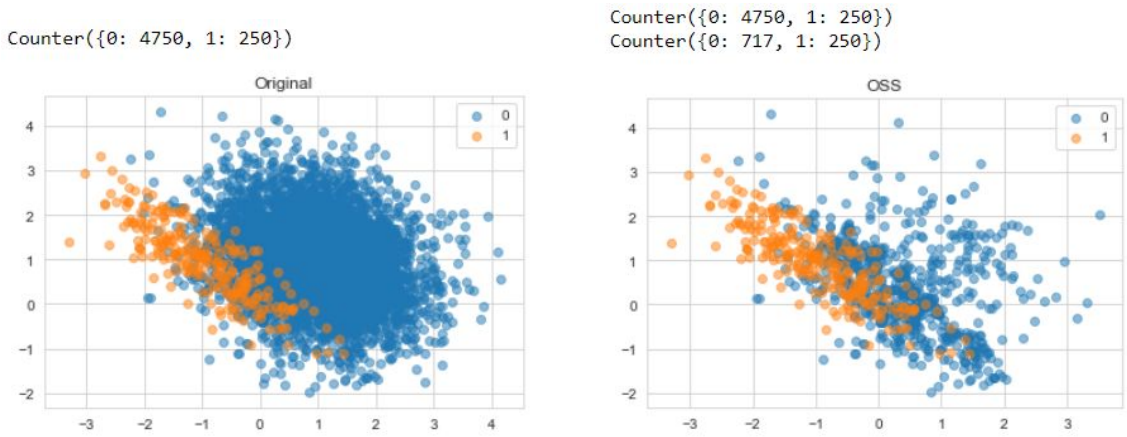

Figure 27: Before and After OSS Undersampling

6.2.5 Neighbourhood Cleaning Rule

NCR for short combines both the CNN as well as ENN approach. First, all the minority class observations are selected. Next, using the ENN approach, obscure majority class examples are removed. Finally, only one round of the CNN approach is applied to ensure the remaining majority class examples which are misclassified against the train set are deleted (Laurikkala 2002).

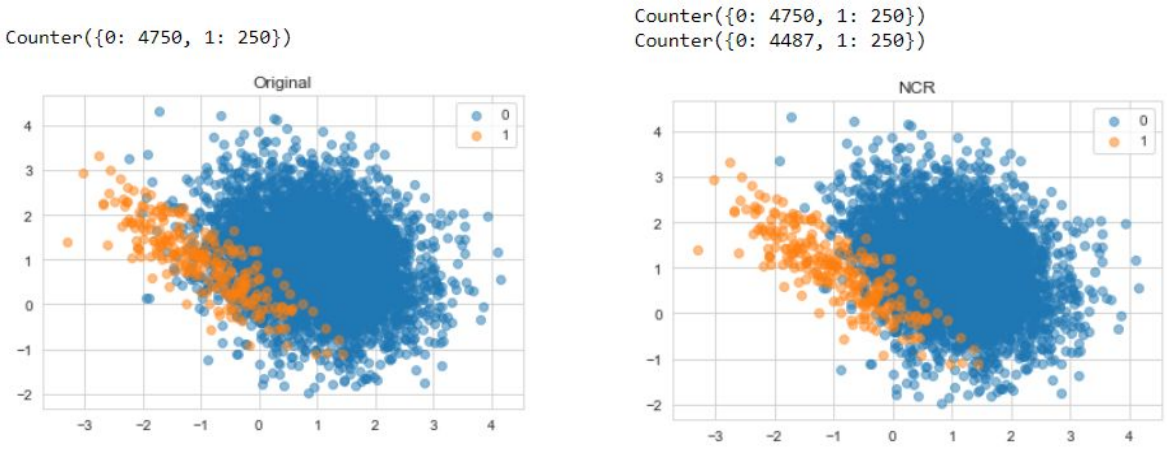

Figure 28: Before and After NCR Undersampling

6.3 Combined Oversampling and Undersampling Methods

SMOTEENN, as mentioned by (Batista et al. 2004), is a technique that combines both the variation of SMOTE oversampling and ENN undersampling. The authors also implemented a technique called SMOTETOMEK, which involved oversampling with SMOTE and undersampling with Tomek links. Although ENN is used to delete examples from both classes, the authors found that using ENN was a lot more effective at deleting examples from the majority class specifically. The synthetic examples below show the effect of SMOTETOMEK and SMOTEENN on an imbalanced dataset.

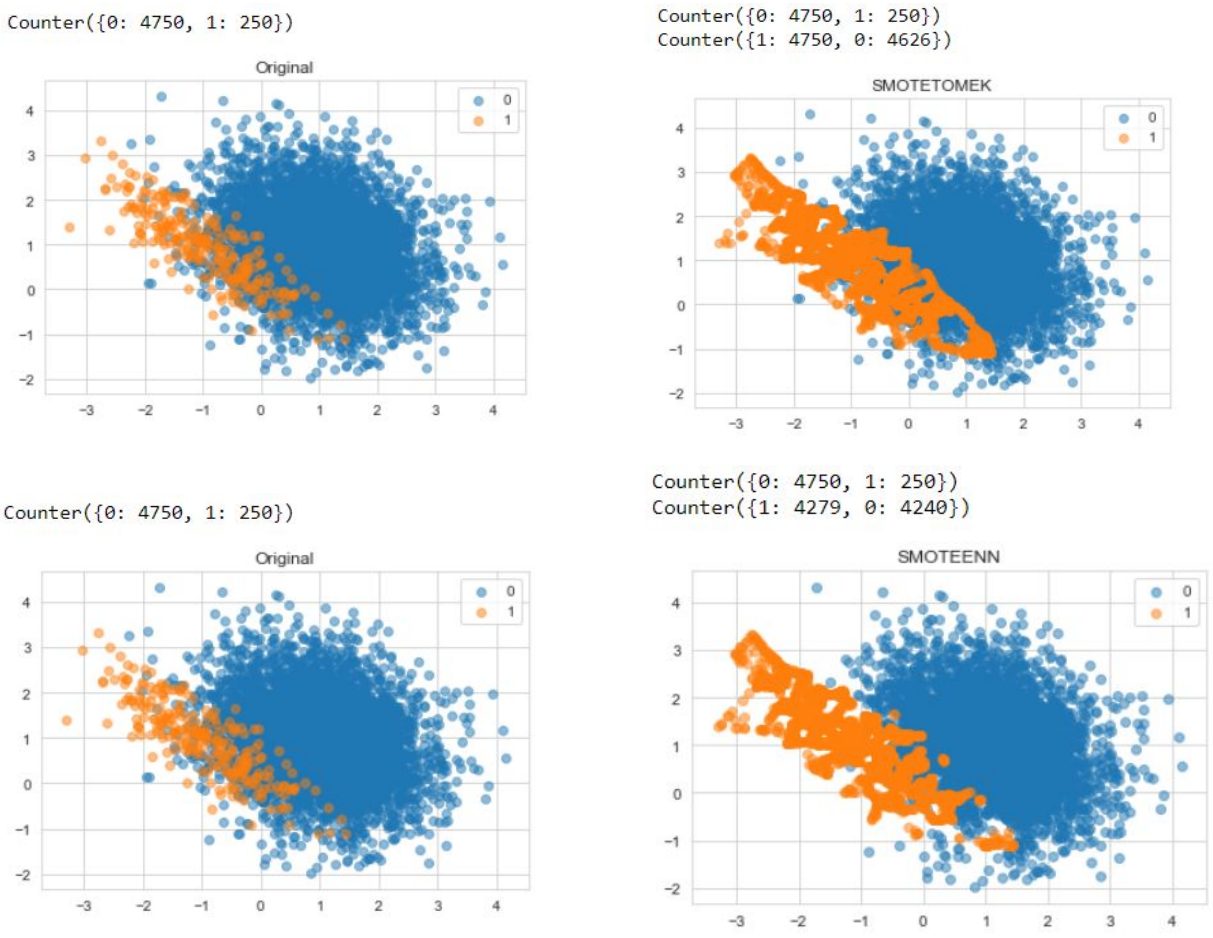

Figure 29: Before and After SMOTETOMEK and SMOTEENN Undersampling

6.4 Cost-Sensitive Algorithms

Most of the time, the focus is to improve the machine learning model performance. There are various ways to do this. One way is to implement cost-sensitive learning. This method focuses on misclassification, which intuitively means that each class has a cost associated with it. The focus is to reduce the misclassification cost in order to achieve better performance. Machine learning models do not know the impact of a misclassification unless told what the costs or penalty associated with them are. In the case of this project, it is highlighted logically and with the issue of a cost matrix initially provided with the data set that false negatives are more costly than false positives. This is typically the case for most imbalanced machine learning problems

and the applications in the real world (Thai-Nghe et al. 2010). The cost matrix provided with the data set emphasized that the cost of a false positive is one and a false negative is five. Usually, a cost matrix could be defined by an economist or an expert in the field. Given the costs above, they can be fed into algorithms that contain a class weight parameter, or the costs can be calculated using the formula below:

$$Weights = \frac{Number\ of\ Samples}{Number\ of\ Classes * Number\ of\ Samples\ Per\ Class}$$

Minority class the weight would be:

$$Weight = \frac{1000}{2 * 700} = 0.71$$

Majority class weight would be:

$$Weight = \frac{1000}{2 * 300} = 1.6$$

This simplifies to a majority to minority ratio weight of 0.71 : 1.60 or 1 : 2.25. Alternatively, this was achieved by setting the class weight parameter for the machine learning models that support this to 'balanced'.

6.5 One-Class Algorithms

6.5.1 One-Class Support Vector Machine Intuition

With a similar intuition of the traditional SVM approach that works with the idea of implementing a decision boundary, The one-class SVM is trained with data that is only from a single class (i.e. the majority class). With this information, the algorithm can identify what is not a 'normal' example and classifies whether the new data point is an outlier or not. It algorithmically functions by identifying the concentrated densities of where the majority class examples tend to be and classifies examples which fall far away from these concentrated spaces as anomalies (Schölkopf et al. 2001).

6.5.2 Isolation Forest Intuition

Decision trees are the basis of isolation forests, and each decision tree is grown randomly. An isolation forest uses a random forest formed out of various decision trees. The algorithm picks a random threshold value and splits the data set in two parts. As the data set gets reduced into smaller subsamples, each observation gets isolated from all the others. It is important to note that in the feature space, anomalies are far away from the other observations; thus they get to be isolated in fewer steps than routine observations (Liu et al. 2012).

6.5.3 Minimum Covariance Determinant Intuition

This technique tends to work well if the input variables follow a Gaussian-like distribution. However, this can also be achieved through a power transformation of the dataset before fitting the model. A spheroid then surrounds the routine observations, and any observations that fall outside the spheroid are classed as outliers (Hubert et al. 2018).

6.5.4 Local Outlier Factor Intuition

In brief, this algorithm also works using a k-nearest neighbour basis. It works by selecting a 'k' value to search for the nearest neighbours. Based on this, each data point is assigned a relevant score to check for the level of isolation, i.e. how far away it is from the surrounding neighbourhood. The higher the score for a given data point, the higher chances it has of being an outlier (Breunig et al. 2000).

Chapter 7: Results

For the prediction of probabilities, it was necessary to attain a baseline performance to which every model could be compared. In previous chapters, it was mentioned that the Brier Skill Score (BSS) would be used. Nevertheless, in order to calculate the BSS, the Brier score had to be calculated first.

The baseline prediction in which probabilities were being predicted was the probability of the positive target class label. The positive class represented roughly 30% of the dataset. Thus,

predicting 0.30 represented the baseline performance for this dataset. If a model had a Brier score greater than 0.30, it represented one with some skill.

Otherwise, the model had no skill. This reference probability could then be used in the calculation for the BSS, and hence any model that had a BSS of greater than 0.000 meant it showed skill on the dataset. The dummy classifier was set to the strategy 'prior' which predicted the prior probability of the given class within each train set.

As such, it was expected that the dummy classifier achieved a baseline BSS of 0.000 with a standard deviation of 0.000 as it was the same as the reference model and also because the test set was being used to evaluate the reference strategy.

In the case of predicting class labels, it was ideal to set the dummy classifier to predict the minority class for all data points within the dataset. This ensured a high recall score adequate for a baseline performer to minimise the false negatives. In such a case, the dummy classifier was set to be the strategy 'constant' with the constant set to be the minority class itself. The dummy classifier achieved a mean baseline F2 score of 0.682 with a standard deviation of 0.000. This meant that any model that achieved scores higher than this presented some skill and models that achieved scores lower than this had no skill on this data set.

Note that only the best performing models with their respective transforms **highlighted in green** are then performed sampling on.

The overall best performing models for the specific cases are **highlighted in yellow.** However, this may not mean that they were better than the dummy classifier performance. For models to be finalised in the end, it was compulsory to have scored better than their respective dummy classifiers.

7.1 Predicting Probabilities - Brier Skill Score

The scores below illustrate the use of standard machine learning algorithms on the data set. It was clear from this that the best performing model was the normalised calibrated Platt scaled nonlinear support vector machine with a mean score of 0.212 and a standard deviation of 0.052. This was better than the dummy classifier that had a mean and standard deviation of 0.000.

Initial Models

	Standardised	Normalised	Norm + Power Transform	Std + Power Transform
LR	0.169 (0.061)	0.203 (0.068)	0.180 (0.068)	0.179 (0.068)
GNB	0.110 (0.072)	0.109 (0.099)	0.084 (0.097)	0.083 (0.098)
LDA	0.164 (0.062)	0.203 (0.073)	0.180 (0.067)	0.180 (0.067)
Linear SVC	0.161 (0.046)	0.200 (0.060)	0.176 (0.049)	0.177 (0.049)
Calibrated Isotonic ET	0.191 (0.053)	0.197 (0.055)	0.171 (0.049)	0.168 (0.052)
Calibrated Platt Scaled ET	0.187 (0.046)	0.195 (0.048)	0.169 (0.043)	0.171 (0.041)
Calibrated Isotonic RF	0.169 (0.059)	0.186 (0.054)	0.164 (0.046)	0.162 (0.051)
Calibrated Platt Scaled RF	0.165 (0.053)	0.184 (0.045)	0.163 (0.042)	0.159 (0.045)
Calibrated Isotonic GBM	0.147 (0.062)	0.170 (0.055)	0.154 (0.049)	0.150 (0.062)
Calibrated Platt Scaled GBM	0.148 (0.050)	0.170 (0.046)	0.157 (0.042)	0.151 (0.048)
Calibrated Isotonic DT	0.040 (0.016)	0.052 (0.021)	0.041 (0.016)	0.042 (0.022)
Calibrated Platt Scaled DT	0.036 (0.013)	0.049 (0.020)	0.039 (0.018)	0.041 (0.020)
Calibrated Isotonic NL SVC	0.162 (0.074)	0.210 (0.065)	0.196 (0.068)	0.194 (0.067)
Calibrated Platt Scaled NL SVC	0.169 (0.058)	0.212 (0.052)	0.202 (0.061)	0.201 (0.061)
Calibrated Isotonic KNN	0.090 (0.056)	0.113 (0.038)	0.087 (0.035)	0.088 (0.036)
Calibrated Platt Scaled KNN	0.090 (0.051)	0.117 (0.033)	0.085 (0.036)	0.085 (0.037)

Figure 30: Brier Skill Score - Standard Machine Learning Algorithms

The figure below illustrates the results of oversampling the best performing algorithms from the ones above. It was clear to see that the best performing model was again the normalised calibrated Platt scaled nonlinear support vector machine SMOTE with a mean score of 0.186 and standard deviation of 0.083. This performed better than the dummy classifier score. However, it seemed to be that oversampling techniques did not achieve a score better than 0.212.

Sampling - Oversample - Using Best Performing Intial Model Data Transform

	ROS	SMOTE	BLSMOTE	SVMSMOTE	ADASYN
LR	0.084 (0.075)	0.098 (0.088)	0.079 (0.085)	0.108 (0.091)	0.102 (0.085)
GNB	-0.002 (0.091)	0.002 (0.113)	0.003 (0.107)	-0.013 (0.107)	0.010 (0.112)
LDA	0.074 (0.084)	0.080 (0.095)	0.064 (0.093)	0.092 (0.095)	0.089 (0.084)
Linear SVC	0.082 (0.074)	0.096 (0.079)	0.084 (0.074)	0.107 (0.083)	0.100 (0.073)
Calibrated Isotonic ET	-0.042 (0.034)	0.145 (0.084)	0.136 (0.081)	0.149 (0.074)	0.142 (0.084)
Calibrated Isotonic RF	0.018 (0.054)	0.167 (0.083)	0.153 (0.091)	0.173 (0.076)	0.159 (0.084)
Calibrated Platt Scaled GBM	0.139 (0.082)	0.130 (0.068)	0.124 (0.065)	0.138 (0.078)	0.129 (0.066)
Calibrated Isotonic DT	0.058 (0.072)	0.034 (0.051)	0.044 (0.045)	0.051 (0.042)	0.037 (0.033)
Calibrated Platt Scaled NL SVC	0.150 (0.088)	0.186 (0.083)	0.179 (0.087)	0.172 (0.084)	0.178 (0.091)
Calibrated Platt Scaled KNN	-0.007 (0.067)	0.028 (0.076)	0.022 (0.065)	0.040 (0.074)	0.033 (0.073)

Figure 31: Brier Skill Score - Oversampling

Next, it was noticeable that the best performing classifier was again the normalised calibrated Platt scaled nonlinear support vector machine TL with a mean score of 0.207 and standard deviation of 0.061. This was better than the results achieved from oversampling but still did not perform better than the initial winning classifier. However, it performed better than the dummy classifier.

Sampling - Undersample - Using Best Performing Intial Model Data Transform

	TL	ENN	RENN	OSS	NCR
LR	0.196 (0.072)	-0.075 (0.109)	-0.371 (0.144)	0.195 (0.072)	-0.016 (0.110)
GNB	0.101 (0.080)	-0.165 (0.137)	-0.454 (0.157)	0.101 (0.081)	-0.119 (0.132)
LDA	0.192 (0.078)	-0.131 (0.118)	-0.436 (0.157)	0.192 (0.078)	-0.080 (0.125)
Linear SVC	0.197 (0.065)	-0.064 (0.098)	-0.357 (0.141)	0.197 (0.064)	-0.006 (0.092)
Calibrated Isotonic ET	0.181 (0.069)	-0.176 (0.124)	-0.558 (0.151)	0.179 (0.063)	-0.075 (0.122)
Calibrated Isotonic RF	0.181 (0.061)	-0.140 (0.126)	-0.521 (0.139)	0.177 (0.064)	-0.050 (0.104)
Calibrated Platt Scaled GBM	0.170 (0.046)	-0.091 (0.091)	-0.432 (0.108)	0.170 (0.047)	-0.009 (0.075)
Calibrated Isotonic DT	0.065 (0.023)	-0.070 (0.062)	-0.333 (0.078)	0.059 (0.020)	-0.053 (0.045)
Calibrated Platt Scaled NL SVC	0.207 (0.061)	-0.142 (0.105)	-0.477 (0.122)	0.206 (0.060)	-0.053 (0.106)
Calibrated Platt Scaled KNN	0.116 (0.040)	-0.259 (0.121)	-0.662 (0.143)	0.114 (0.041)	-0.120 (0.087)

Figure 32: Brier Skill Score - Undersampling

Coming onto combined sampling approaches, the image below illustrates the results, and it was noticeable that the normalised calibrated Platt scaled nonlinear support vector machine SMOTETOMEK was again the best performer under this section with a mean of 0.187 and

standard deviation of 0.088. This still does not beat the initial score of 0.212. However, it performed better than the dummy classifier.

Sampling - Combined - Using Best Performing Intial Model Data Transform

	SMOTEENN	SMOTETOMEK
LR	-0.445 (0.126)	0.096 (0.088)
GNB	-0.546 (0.170)	0.005 (0.118)
LDA	-0.524 (0.154)	0.070 (0.090)
Linear SVC	-0.441 (0.134)	0.081 (0.083)
Calibrated Isotonic ET	-0.074 (0.143)	0.145 (0.070)
Calibrated Isotonic RF	-0.177 (0.140)	0.164 (0.078)
Calibrated Platt Scaled GBM	-0.354 (0.108)	0.120 (0.075)
Calibrated Isotonic DT	-0.416 (0.116)	0.054 (0.047)
Calibrated Platt Scaled NL SVC	-0.314 (0.129)	0.187 (0.088)
Calibrated Platt Scaled KNN	-0.655 (0.158)	0.030 (0.079)

Figure 33: Brier Skill Score - Combined Sampling

The final assessment was using cost-sensitive algorithms. Only some of the algorithms from above contained a class weight parameter. Thus, only those were used. Using a balanced class weight, the results showed that the standardised calibrated Platt scaled extra trees algorithm achieved a mean score of 0.223 and a standard deviation of 0.058. This score was much better than the initial score achieved in the first part of 0.212 as well as the dummy classifier.

Cost Sensitive Models - Balanced Class Weights

	Standardised	Normalised	Norm + Power Transform	Std + Power Transform
LR	0.100 (0.070)	0.068 (0.087)	0.098 (0.079)	0.098 (0.079)
Linear SVC	0.209 (0.055)	0.179 (0.058)	0.197 (0.056)	0.198 (0.056)
Calibrated Isotonic ET	0.222 (0.068)	0.201 (0.058)	0.202 (0.052)	0.203 (0.049)
Calibrated Platt Scaled ET	0.223 (0.058)	0.203 (0.050)	0.203 (0.044)	0.201 (0.040)
Calibrated Isotonic RF	0.209 (0.079)	0.190 (0.053)	0.205 (0.053)	0.204 (0.058)
Calibrated Platt Scaled RF	0.207 (0.065)	0.194 (0.045)	0.203 (0.048)	0.202 (0.048)
Calibrated Isotonic DT	0.063 (0.027)	0.057 (0.016)	0.060 (0.019)	0.056 (0.025)
Calibrated Platt Scaled DT	0.060 (0.028)	0.053 (0.018)	0.057 (0.018)	0.054 (0.022)
Calibrated Isotonic NL SVC	0.218 (0.070)	0.197 (0.070)	0.202 (0.075)	0.202 (0.074)
Calibrated Platt Scaled NL SVC	0.221 (0.057)	0.200 (0.056)	0.209 (0.061)	0.208 (0.061)

Figure 34: Brier Skill Score - Cost Sensitive - Balanced Class Weights

Moreover, testing the given class weights from the cost matrix was also necessary for comparative purposes. The results below showed that the standardised calibrated Platt scaled extra trees algorithm achieved a mean score of 0.220 and a standard deviation of 0.054. However, this score did not surpass the previous high of 0.223, but the model performed better than the dummy classifier.

Cost Sensitive Models - Given Class Weights

	Standardised	Normalised	Norm + Power Transform	Std + Power Transform
LR	-0.173 (0.087)	-0.217 (0.094)	-0.162 (0.095)	-0.163 (0.095)
Linear SVC	0.190 (0.052)	0.151 (0.056)	0.184 (0.049)	0.184 (0.049)
Calibrated Isotonic ET	0.217 (0.068)	0.201 (0.056)	0.204 (0.052)	0.200 (0.056)
Calibrated Platt Scaled ET	0.220 (0.054)	0.205 (0.050)	0.201 (0.048)	0.202 (0.049)
Calibrated Isotonic RF	0.187 (0.077)	0.182 (0.050)	0.181 (0.070)	0.184 (0.066)
Calibrated Platt Scaled RF	0.191 (0.064)	0.184 (0.045)	0.183 (0.061)	0.182 (0.060)
Calibrated Isotonic DT	0.057 (0.023)	0.055 (0.018)	0.052 (0.023)	0.056 (0.026)
Calibrated Platt Scaled DT	0.057 (0.021)	0.055 (0.016)	0.051 (0.025)	0.057 (0.026)
Calibrated Isotonic NL SVC	0.188 (0.076)	0.165 (0.070)	0.172 (0.076)	0.172 (0.076)
Calibrated Platt Scaled NL SVC	0.193 (0.062)	0.167 (0.055)	0.180 (0.062)	0.179 (0.062)

Figure 35: Brier Skill Score - Cost Sensitive - Given Class Weights

The overall performances could be concluded with the summary table below showing that with the use of the Brier Skill Score, if predicting probabilities was the key goal, then it would be best to use the best overall performing algorithm which was the standardised calibrated Platt scaled extra trees with balanced class weights alongside the appropriate data preparation and cross-validation techniques stated in Chapters 2 and 4. This algorithm also had a score higher than the dummy classifier, which made it ideal to use.

Best Algorithms	Transform	Strategy	Brier Skill Score
Calibrated Platt Scaled NL SVC	Normalised	N/A	0.212 (0.052)
Calibrated Platt Scaled NL SVC	Normalised	SMOTE	0.186 (0.083)
Calibrated Platt Scaled NL SVC	Normalised	TL	0.207 (0.061)
Calibrated Platt Scaled NL SVC	Normalised	SMOTETOMEK	0.187 (0.088)
Calibrated Platt Scaled ET	Standardised	BALANCED CW	0.223 (0.058)
Calibrated Platt Scaled ET	Standardised	GIVEN CW	0.220 (0.054)

Figure 36: Brier Skill Score - Final Verdict

Probability predictions for being a good client were made using the best model highlighted from the image above on new data and the results are shown below:

```
Good Customers:
>data=['A11', 6, 'A34', 'A43', 1169, 'A65', 'A75', 4, 'A93', 'A101', 4, 'A121', 67, 'A143', 'A152', 2, 'A173', 1, 'A192', 'A20
1'], Good=95.419%
>data=['A14', 12, 'A34', 'A46', 2096, 'A61', 'A74', 2, 'A93', 'A101', 3, 'A121', 49, 'A143', 'A152', 1, 'A172', 2, 'A191', 'A20
1'], Good=95.688%
>data=['A11', 42, 'A32', 'A42', 7882, 'A61', 'A74', 2, 'A93', 'A103', 4, 'A122', 45, 'A143', 'A153', 1, 'A173', 2, 'A191', 'A20
1'], Good=89.521%
---------------------------------------
Bad Customers:
>data=['A13', 18, 'A32', 'A43', 2100, 'A61', 'A73', 4, 'A93', 'A102', 2, 'A121', 37, 'A142', 'A152', 1, 'A173', 1, 'A191', 'A20
1'], Good=5.180%
>data=['A11', 24, 'A33', 'A40', 4870, 'A61', 'A73', 3, 'A93', 'A101', 4, 'A124', 53, 'A143', 'A153', 2, 'A173', 2, 'A191', 'A20
1'], Good=6.012%
>data=['A11', 24, 'A32', 'A43', 1282, 'A62', 'A73', 4, 'A92', 'A101', 2, 'A123', 32, 'A143', 'A152', 1, 'A172', 1, 'A191', 'A20
1'], Good=5.180%
```

Figure 37: New Predictions - Probabilities

For the cases of good clients, the probability of being good was high, between 89% and 96%. However, for the bad clients, the probability of being good ranges from 5% to 6%.

7.2 Predicting Class Labels - F2 Score

The scores below illustrate the use of standard machine learning algorithms on the data set. It was clear from this that the best performing model was the normalised LDA with a mean score of 0.500 and a standard deviation of 0.073. However, this was not better than the dummy classifier that had a mean of 0.682 and a standard deviation of 0.000.

Initial Models

	Standardised	Normalised	Norm + Power Transform	Std + Power Transform
LR	0.396 (0.079)	0.476 (0.070)	0.462 (0.075)	0.462 (0.076)
GNB	0.424 (0.074)	0.417 (0.085)	0.452 (0.078)	0.452 (0.080)
LDA	0.410 (0.080)	0.500 (0.073)	0.460 (0.082)	0.462 (0.079)
Linear SVC	0.374 (0.076)	0.484 (0.067)	0.474 (0.077)	0.465 (0.079)
ET	0.197 (0.083)	0.260 (0.067)	0.214 (0.075)	0.221 (0.073)
RF	0.243 (0.076)	0.286 (0.084)	0.256 (0.076)	0.257 (0.092)
GBM	0.346 (0.084)	0.419 (0.076)	0.405 (0.079)	0.382 (0.099)
DT	0.412 (0.077)	0.439 (0.081)	0.411 (0.090)	0.417 (0.086)
NL SVC	0.313 (0.088)	0.429 (0.074)	0.411 (0.090)	0.411 (0.092)
KNN	0.363 (0.078)	0.436 (0.077)	0.402 (0.103)	0.401 (0.100)

Figure 38: F2 Score - Standard Machine Learning Algorithms

The figure below illustrates the results of oversampling the best performing algorithms from the ones above. It was clear to see that the best performing model was the normalised LDA along with a random oversampling technique that provided a mean score of 0.685 and standard deviation of 0.061. This algorithm also performed better than the dummy classifier.

Sampling - Oversample - Using Best Performing Intial Model Data Transform

	ROS	SMOTE	BLSMOTE	SVMSMOTE	ADASYN
LR	0.661 (0.060)	0.662 (0.068)	0.673 (0.063)	0.657 (0.071)	0.664 (0.064)
GNB	0.637 (0.076)	0.566 (0.077)	0.573 (0.068)	0.618 (0.076)	0.570 (0.069)
LDA	0.685 (0.061)	0.663 (0.073)	0.674 (0.069)	0.673 (0.077)	0.667 (0.068)
Linear SVC	0.667 (0.064)	0.664 (0.073)	0.683 (0.060)	0.662 (0.071)	0.666 (0.064)
ET	0.225 (0.055)	0.453 (0.057)	0.458 (0.075)	0.435 (0.057)	0.440 (0.074)
RF	0.411 (0.058)	0.494 (0.066)	0.480 (0.074)	0.525 (0.082)	0.498 (0.058)
GBM	0.552 (0.070)	0.566 (0.065)	0.576 (0.059)	0.589 (0.083)	0.572 (0.063)
DT	0.406 (0.080)	0.480 (0.058)	0.474 (0.093)	0.476 (0.083)	0.482 (0.073)
NL SVC	0.591 (0.063)	0.561 (0.068)	0.565 (0.069)	0.591 (0.064)	0.569 (0.066)
KNN	0.616 (0.036)	0.645 (0.058)	0.654 (0.047)	0.619 (0.059)	0.629 (0.049)

Figure 39: F2 Score - Oversampling

Next, it was noticeable from the image below that the best performing classifier was the normalised extra trees RENN with a mean score of 0.721 and standard deviation of 0.037. This was better than the results achieved from oversampling and initial models and even better than the performance of the dummy classifier.

Sampling - Undersample - Using Best Performing Intial Model Data Transform

	TL	ENN	RENN	OSS	NCR
LR	0.528 (0.071)	0.697 (0.048)	0.720 (0.052)	0.528 (0.070)	0.685 (0.056)
GNB	0.509 (0.079)	0.655 (0.058)	0.697 (0.043)	0.514 (0.077)	0.659 (0.062)
LDA	0.541 (0.070)	0.706 (0.046)	0.718 (0.050)	0.540 (0.068)	0.687 (0.057)
Linear SVC	0.545 (0.067)	0.685 (0.047)	0.714 (0.048)	0.544 (0.062)	0.671 (0.059)
ET	0.322 (0.051)	0.676 (0.055)	0.721 (0.037)	0.334 (0.076)	0.646 (0.064)
RF	0.348 (0.070)	0.678 (0.049)	0.718 (0.037)	0.345 (0.076)	0.650 (0.072)
GBM	0.455 (0.061)	0.674 (0.052)	0.703 (0.039)	0.450 (0.073)	0.651 (0.069)
DT	0.431 (0.069)	0.597 (0.070)	0.649 (0.060)	0.424 (0.078)	0.577 (0.070)
NL SVC	0.508 (0.067)	0.679 (0.048)	0.707 (0.040)	0.511 (0.065)	0.651 (0.063)
KNN	0.485 (0.064)	0.652 (0.054)	0.672 (0.051)	0.485 (0.065)	0.648 (0.056)

Figure 40: F2 Score - Undersampling

Moving on to combined sampling approaches, normalised logistic regression SMOTEENN appeared to be the best from the range and achieved a performing mean score of 0.732 and a

standard deviation of 0.039. This score was better than the ones achieved from all of the above as well as the dummy classifier.

Sampling - Combined - Using Best Performing Intial Model Data Transform

	SMOTEENN	SMOTETOMEK
LR	0.732 (0.039)	0.662 (0.063)
GNB	0.685 (0.056)	0.571 (0.066)
LDA	0.728 (0.047)	0.673 (0.065)
Linear SVC	0.726 (0.038)	0.663 (0.070)
ET	0.714 (0.034)	0.457 (0.065)
RF	0.715 (0.030)	0.515 (0.071)
GBM	0.713 (0.031)	0.573 (0.064)
DT	0.629 (0.054)	0.482 (0.068)
NL SVC	0.706 (0.040)	0.566 (0.069)
KNN	0.695 (0.039)	0.646 (0.055)

Figure 41: F2 Score - Combined Sampling

Utilising cost-sensitive approaches, it was noticeable that when using the balanced class weight parameter, the normalised and power transformation version of logistic regression was the best in its range and achieved a mean score of 0.670 and a standard deviation of 0.056. Although the score was better than the initial models, it did not beat the dummy classifier.

Cost Sensitive Models - Balanced Class Weights

	Standardised	Normalised	Norm + Power Transform	Std + Power Transform
LR	0.657 (0.057)	0.667 (0.049)	0.670 (0.056)	0.669 (0.054)
Linear SVC	0.663 (0.054)	0.668 (0.053)	0.665 (0.057)	0.663 (0.060)
ET	0.186 (0.081)	0.219 (0.065)	0.192 (0.074)	0.196 (0.074)
RF	0.230 (0.065)	0.237 (0.054)	0.242 (0.075)	0.219 (0.076)
DT	0.451 (0.094)	0.413 (0.062)	0.427 (0.080)	0.408 (0.074)
NL SVC	0.653 (0.067)	0.625 (0.067)	0.616 (0.074)	0.616 (0.072)

Figure 42: F2 Score - Cost Sensitive - Balanced Class Weights

However, a massive change is seen when utilising the given class weights associated with the cost matrix. A standardised logistic regression model appeared to be the best and achieved a

mean score of 0.722, with a standard deviation of 0.043. This performed better than the dummy classifier and the balanced weights version above.

Cost Sensitive Models - Given Class Weights

	Standardised	Normalised	Norm + Power Transform	Std + Power Transform
LR	0.722 (0.043)	0.697 (0.045)	0.715 (0.042)	0.715 (0.043)
Linear SVC	0.714 (0.039)	0.706 (0.052)	0.716 (0.037)	0.715 (0.036)
ET	0.156 (0.062)	0.207 (0.070)	0.181 (0.080)	0.178 (0.069)
RF	0.224 (0.064)	0.237 (0.066)	0.225 (0.076)	0.233 (0.071)
DT	0.444 (0.083)	0.387 (0.091)	0.424 (0.088)	0.417 (0.081)
NL SVC	0.658 (0.064)	0.634 (0.068)	0.643 (0.065)	0.643 (0.066)

Figure 43: F2 Score - Cost Sensitive - Given Class Weights.

Finally, examining the anomaly detection algorithms below. It was clear that the standardised local outlier factor achieved the best mean score from its other competing algorithms of the same category. It achieved a mean score of 0.409 and standard deviation of 0.080. However, it did not beat the dummy classifier performance and was the lowest-performing 'best' algorithm.

Initial Models

	Standardised	Normalised	Norm + Power Transform	Std + Power Transform
One-Class SVM	0.405 (0.084)	0.260 (0.068)	0.365 (0.085)	0.364 (0.084)
Isolation Forest	0.406 (0.068)	0.362 (0.086)	0.384 (0.087)	0.393 (0.087)
Minimum Covariance Determinant	0.372 (0.081)	0.369 (0.089)	0.358 (0.088)	0.369 (0.097)
Local Outlier Factor	0.409 (0.080)	0.326 (0.079)	0.346 (0.080)	0.347 (0.081)

Figure 44: F2 Score - Anomaly Detection

The overall performances could be concluded with the summary table below showing that with the use of the F2 Score, if predicting class labels was the key goal, then it would be best to use the best overall performing algorithm which was the normalised logistic regression model with a SMOTEENN sampling strategy alongside the appropriate data preparation and cross-validation

techniques stated in Chapters 2 and 4. This algorithm also had a score higher than the dummy classifier, which made it ideal to use.

Best Algorithms	Transform	Strategy	F2 Score
LDA	Normalised	N/A	0.500 (0.073)
LDA	Normalised	ROS	0.685 (0.061)
ET	Normalised	RENN	0.721 (0.037)
LR	Normalised	SMOTEENN	0.732 (0.039)
LR	Normalised + Power Transform	BALANCED CW	0.670 (0.056)
LR	Standardised	GIVEN CW	0.722 (0.043)
LOF	Standardised	N/A	0.409 (0.080)

Figure 45: F2 Score - Verdict

Class label predictions for being a good client using new rows of data were made using the best model from the image above and the results are shown below:

```
Good Customers:
>data=['A11', 6, 'A34', 'A43', 1169, 'A65', 'A75', 4, 'A93', 'A101', 4, 'A121', 67, 'A143', 'A152', 2, 'A173', 1, 'A192', 'A20
1'], Predicted=0 (expected 0)
>data=['A14', 12, 'A34', 'A46', 2096, 'A61', 'A74', 2, 'A93', 'A101', 3, 'A121', 49, 'A143', 'A152', 1, 'A172', 2, 'A191', 'A20
1'], Predicted=0 (expected 0)
>data=['A11', 42, 'A32', 'A42', 7882, 'A61', 'A74', 2, 'A93', 'A103', 4, 'A122', 45, 'A143', 'A153', 1, 'A173', 2, 'A191', 'A20
1'], Predicted=1 (expected 0)
----------------------------------------
Bad Customers:
>data=['A13', 18, 'A32', 'A43', 2100, 'A61', 'A73', 4, 'A93', 'A102', 2, 'A121', 37, 'A142', 'A152', 1, 'A173', 1, 'A191', 'A20
1'], Predicted=1 (expected 1)
>data=['A11', 24, 'A33', 'A40', 4870, 'A61', 'A73', 3, 'A93', 'A101', 4, 'A124', 53, 'A143', 'A153', 2, 'A173', 2, 'A191', 'A20
1'], Predicted=1 (expected 1)
>data=['A11', 24, 'A32', 'A43', 1282, 'A62', 'A73', 4, 'A92', 'A101', 2, 'A123', 32, 'A143', 'A152', 1, 'A172', 1, 'A191', 'A20
1'], Predicted=1 (expected 1)
```

Figure 46: New Predictions - Class Labels

Most cases of good clients were correctly predicted. However, the model did make a mistake on the third row emphasising that the model is not perfect. It was pleasing to see that all cases of bad clients were correctly predicted as bad.

Chapter 8: Conclusion and Future Works

Given that there was quite a bit of variance within the results. It seemed that supervised learning algorithms had better results than unsupervised anomaly detection methods. Even though these supervised algorithms proved to have skill on this dataset, it did not mean that they attained the best results possible.

Change of metrics could have brought about different results depending on what the business required. For example, if both probabilities and class labels were required from the same algorithm and the positive class was the focus, then the area under the precision-recall curve (PR-AUC) would have been an ideal metric to use. If both classes were equally crucial according to business needs, then the area under the receiver operating curve (ROC-AUC) would have been ideal. Another case would be if class labels were required, and both classes were equally important, then the geometric mean would have been the right choice (G-Mean).

In an attempt to attain the best results, one could implement various other combinations of combined sampling strategies. More sampling strategies, such as the near-miss approach or condensed nearest neighbours (CNN), could also be tested. Another idea would be to perform the various sampling strategies along with cost-sensitive models using the balanced and given class weights.

One could go ahead and perform hyperparameter tuning on all algorithms or the selected final algorithms using a grid search or randomised search approach if there are no limits on computational resources. A quicker way would be to use Bayesian optimisation, which is known to be fast and effective.

It would also be useful to test out other feature extraction techniques such as singular value decomposition, iso map embedding, locally linear embedding and modified locally linear embedding. With sufficient computational power, better feature engineering could have also been done, e.g. creating a feature union data frame consisting of features from a feature

extraction algorithm as well as a feature selection method such as recursive feature elimination with inbuilt cross-validation. These techniques could also be tested with various combinations of feature extraction and feature selection methods, including feature importances which is provided with most ensemble algorithms.

Probability threshold moving could also be implemented within the pipeline procedure to find the optimal probability threshold of prediction. The default threshold was set to 0.5.

A discretisation of continuous features could also be performed to make them categorical, e.g. uniform discretisation, k-means discretisation, quantile discretisation. In addition to this, different categorical encodings could also be applied, such as ordinal encoder for variables that had an ordered sequence, e.g. 'Savings' or 'Employment'. New input variables could also be derived using a polynomial feature transform and adjusting the effect of polynomial degree.

The project utilised the Yeo-Johnson power transformation to make variable distributions Gaussian-like as some machine learning models perform better with such a distribution as stated earlier. However, another option to change the distribution of input data would be to use a quantile transform with either a uniform or normal quantile transform.

Other variations of data splits could be implemented, such as leave one out cross-validation (LOOCV), repeated random train test splits and nested cross-validation. Moreover, testing out a variety of contamination parameters for anomaly detection methods and testing deep learning methods could also be implemented. Besides, the results from the machine learning algorithms could also be compared against the credit score of a given client to ensure further reliability of results.

Most of the focus in machine learning is awarded to data preparation, and this is rightfully so. The recommendations outlined mainly focus on various data preparation methods rather than utilising more complex models. The reason for this is stated by (Kuhn and Johnson 2013). The

authors mention that the data fed into a predictive model can make or break its performance as different models react differently to the type of input being fed.

Thus it is imperative to focus more on better data preparation methods. This is strengthened by a quote written by Nick Harkaway, which says 'Garbage in, garbage out.' suggesting that low-quality input will most likely lead to poor output.

Bibliography

Albon, C. 2018. *Machine Learning with Python Cookbook Practical Solutions from Preprocessing to Deep Learning*. Cambridge O'reilly.

Allibhai, E. 2018. Holdout vs. Cross-validation in Machine Learning. Available at: https://medium.com/@eijaz/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f#:~:text=Cross%2Dvalidation%20is%20usually%20the [Accessed: 17 July 2020].

Amadeo, K. 2020. Causes of the 2008 Global Financial Crisis. Available at: https://www.thebalance.com/what-caused-2008-global-financial-crisis-3306176#:~:text=Deregulation%20in%20the%20financial%20industry%20was%20the [Accessed: 11 September 2020].

Aurélien Géron 2019. *Hands-on Machine Learning with Scikit-Learn and TensorFlow concepts, tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc.

Baesens, B. et al. 2003. Benchmarking state-of-the-art Classification Algorithms for Credit Scoring. *Journal of the Operational Research Society* 54(6), pp. 627–635. doi: 10.1057/palgrave.jors.2601545.

Batista, G. et al. 2004. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explorations Newsletter* 6(1), pp. 20–29. doi: 10.1145/1007730.1007735.

Bishop, C. 2013. *Pattern Recognition and Machine Learning*. S.L.: Springer, India, Private Ltd.

Bishop, C.M. 2002. *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.

Branco, P. et al. 2016. A Survey of Predictive Modeling on Imbalanced Domains. *ACM Computing Surveys* 49(2). doi: 10.1145/2907070.

Breiman, L. 2001. Random Forests. *Machine Learning* 45(1), pp. 5–32. doi: 10.1023/a:1010933404324.

Breunig, M.M. et al. 2000. LOF: Identifying Density-Based Local Outliers. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data - SIGMOD '00* 29, pp. 93–104. doi: 10.1145/342009.335388.

Chawla, N.V. et al. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16, pp. 321–357. doi: 10.1613/jair.953.

Desai, V.S. et al. 1996. A Comparison of Neural Networks and Linear Scoring Models in the Credit Union Environment. *European Journal of Operational Research* 95(1), pp. 24–37. doi: 10.1016/0377-2217(95)00246-4.

Fernández, A. et al. 2018. *Learning from Imbalanced Data Sets*. Cham, Switzerland: Springer.

Geurts, P. et al. 2006. Extremely Randomized Trees. *Machine Learning* 63(1), pp. 3–42. Available at: https://orbi.uliege.be/bitstream/2268/9357/1/geurts-mlj-advance.pdf [Accessed: 18 June 2019].

Han, H. et al. 2005. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. *Lecture Notes in Computer Science* 3644(1), pp. 878–887. doi: 10.1007/11538059_91.

He, H. et al. 2008. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. *IEEE Xplore* , pp. 1322–1328. doi: 10.1109/IJCNN.2008.4633969.

He, H. et al. 2013. *Imbalanced Learning : foundations, algorithms, and Applications*. Hoboken: John Wiley & Sons, Cop.

Hubert, M. et al. 2018. Minimum Covariance Determinant and Extensions. *WIREs Computational Statistics* 10(3), p. e:1421. doi: 10.1002/wics.1421.

Kubat, M. and Matwin, S. 2000. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. *citeseerx.ist.psu.edu*

Kuhn, M. and Johnson, K. 2013. *Applied Predictive Modeling*. Springer Nature.

Laurikkala, J. 2002. Instance-based Data Reduction for Improved Identification of Difficult Small Classes. *Intelligent Data Analysis* 6(4), pp. 311–322. doi: 10.3233/ida-2002-6402.

Liu, F.T. et al. 2012. Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data* 6(1), pp. 1–39. doi: 10.1145/2133360.2133363.

More, A. 2016. Survey of Resampling Techniques for Improving Classification Performance in Unbalanced Datasets. *arXiv:1608.06048 [cs, stat]* . Available at: https://arxiv.org/abs/1608.06048 [Accessed: 12 August 2020].

Murphy, K.P. and Massachusetts Institute Of Technology 2012. *Machine Learning : A Probabilistic Perspective*. Cambridge (Ma): Mit Press.

Nguyen, H.M. et al. 2011. Borderline over-sampling for Imbalanced Data Classification. *International Journal of Knowledge Engineering and Soft Data Paradigms* 3(1), pp. 4–21. doi: 10.1504/IJKESDP.2011.039875.

Olson, D.L. 2005. Data Set Balancing. *Data Mining and Knowledge Management* 3327, pp. 71–80. doi: 10.1007/978-3-540-30537-8_8.

Raschka, S. and Olson, R.S. 2015. *Python Machine Learning : Unlock Deeper Insights into Machine Learning with This Vital Guide to cutting-edge Predictive Analytics*. Birmingham, Uk: Packt Publishing.

Schölkopf, B. et al. 2001. Estimating the Support of a High-Dimensional Distribution. *Neural Computation* 13(7), pp. 1443–1471. doi: 10.1162/089976601750264965.

Starmer, J. 2019. Gradient Boost Part 3: Classification. *YouTube* . Available at: https://www.youtube.com/watch?v=jxuNLH5dXCs [Accessed: 10 August 2020].

Thai-Nghe, N. et al. 2010. Cost-sensitive Learning Methods for Imbalanced Data. *IEEE Xplore* , pp. 1–8. doi: 10.1109/IJCNN.2010.5596486.

Tomek, I. 1976. Two Modifications of CNN. *Semantic Scholar* SMC-6(11), pp. 769–772. doi: 10.1109/TSMC.1976.4309452.

Venkata, S. 2018. Overview of Cross Validation. Available at: https://medium.com/@venkatasujit272/overview-of-cross-validation-3785d5414ece [Accessed: 17 July 2020].

Weiss, G.M. and Provost, F. 2003. Learning When Training Data Are Costly: the Effect of Class Distribution on Tree Induction. *Journal of Artificial Intelligence Research* 19(1), pp. 315–354. doi: 10.1613/jair.1199.

Wilson, D.L. 1972. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-2(3), pp. 408–421. doi: 10.1109/TSMC.1972.4309137.

Witten, I.H. and Al, E. 2017. *Data Mining : Practical Machine Learning Tools and Techniques*. Amsterdam: Morgan Kaufmann.

Yobas, M.B. et al. 2000. Credit Scoring Using Neural and Evolutionary Techniques. *IMA Journal of Management Mathematics* 11(2), pp. 111–125. doi: 10.1093/imaman/11.2.111.