

**Tom McClelland – 100339002**

**Data Mining: KDD Report**

**Knowledge Discovery from Data**

**Lending Club Loan Dataset**

1. **Abstract**
2. **Introduction**
3. **Supervised Learning**
   1. **Problem Specification**
   2. **Resourcing**
   3. **Data Cleaning**
   4. **Pre-Processing**
   5. **Supervised Data Mining**
   6. **Supervised Evaluation**
4. **Unsupervised Learning**
   1. **Unsupervised Data Mining**
   2. **Unsupervised Evaluation**
5. **Conclusion**
6. **Bibliography**

**0.0 Abstract**

This report covers the processes that were carried out on the Lending Club Loans Dataset to train and evaluate multiple supervised and unsupervised learning algorithms. The dataset goes through a cleaning, pre-processing, mining and evaluation stage, each of which having multiple processes such as feature selection, scaling, sampling etc. The supervised learning algorithms performed well when compared to the unsupervised learning algorithm, but more work is required to measure model performance when the algorithm is only supplied with features that can be provided prior to making a loan agreement.

**1.0 Introduction**

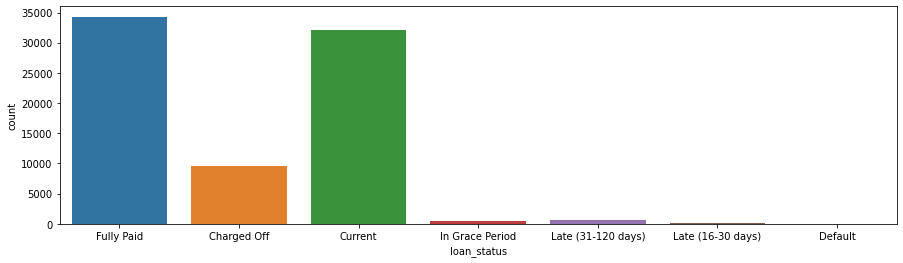
Knowledge discovery from data (KDD) has become an important part of business operations, especially within large businesses that acquire huge amounts of data every minute. While new businesses can sometimes lack a great volume of data, modern procedures and Internet of Things (IoT) devices can allow for these new businesses to gather data at a rapid velocity by incorporating customers as part of their system and using tiny sensors to collect data (Choi et al., 2021). Businesses will then use this data to create pieces of information about their customers followed by aquiring knowledge from the information, and finally by making decisions based on this knowledge. However, these decisions are only as good as the data that they are based on, and in many cases the quality of this data is insufficient. Poor quality data can be very costly to businesses with a piece of research from IBM and Harvard estimating that bad data would have cost the US economy over $3T (Redman, 2016) in 2016. Bad data can occur due to, but not limited to, poor data collection procedures, inconsiderate staff or malfunctioning IoT equipment. While there are many computational approaches to remove bad data and prevent it from influencing decisions (Venayagamoorthy, 2014), there is still the cost of lost opportunity to consider, as one small piece of missing data could result in having to discard numerous more. This report will focus on the large amount of data that money lenders collect about their customers, both prior to and during the time they loan them money. The ‘LendingClubLoans 2018-2020 Database’ (LCLDB) will be used to carry out almost all of the KDD processes, resulting in multiple evaluated classification models that lenders could deploy to predict which customers are high risk.

**2.0 Supervised Learning**

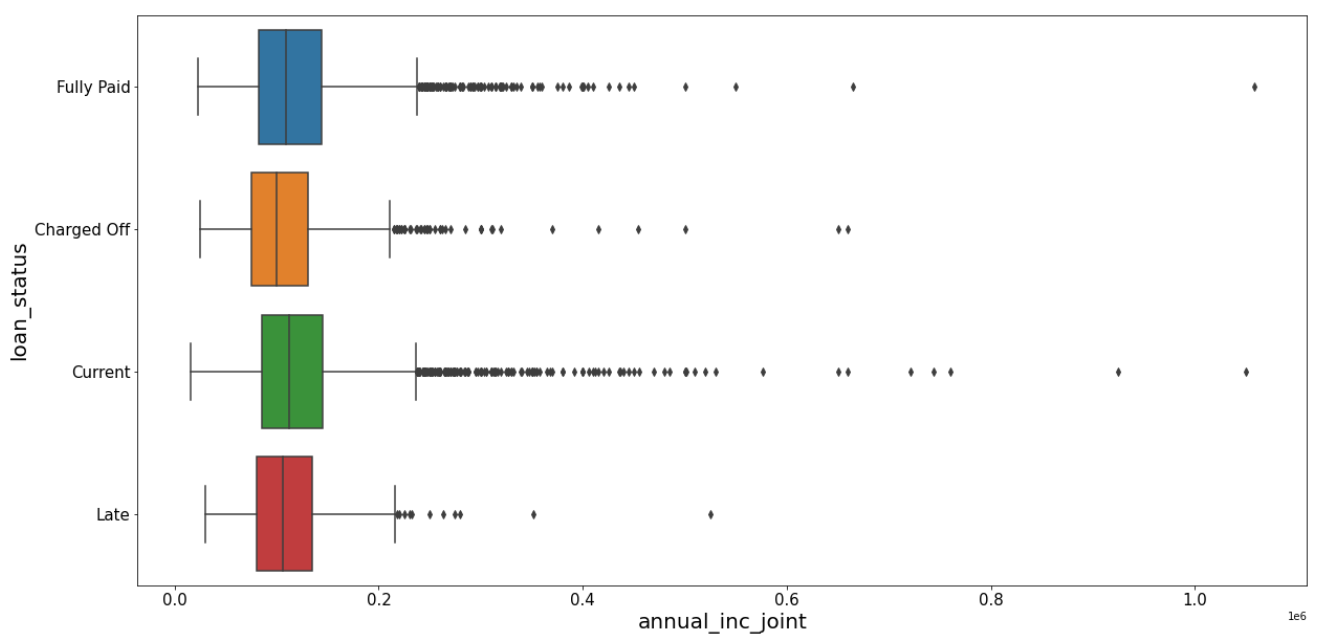
**2.1 Problem Specification**

The first step in the KDD process is problem specification, which will result in a tightly defined problem that the full KDD process will attempt to solve. Fortunately, much of the problem specification work had already been done prior to accessing the database as it came with a basic data dictionary attached to the database and the high/low level tasks were already defined by the assignment brief. A data dictionary has been defined as a list describing the variables, definitions, and attributes of data that have been or are going to be collected in a separate database, the purpose of which is to ensure consistent terminology (AHIMA e-HIM Workgroup on EHR Data Content, 2006). Further benefits of a data include improved data quality, integrity and comparability over time (McCabe et al., 2019). The high level task was to develop a loan status classifier, while the low level tasks included data cleansing, pre-processing and feature summarisation. Software requirements were met by downloading all necessary software prior to the start of the data mining module while hardware requirements were met by using a powerful personal laptop. This left database examination, database familiarisation and a feasibility analysis to be carried out. Examination and familiarisation were carried out simultaneously with the low level task of producing a feature summarisation which included data types, number of missing (null) values and number of unique values for each feature. This feature summarisation was then linked with the data dictionary that had been provided to create a more informative data dictionary, as is shown from a small preview in figure 1. The full dictionary can be seen from the attached Jupyter Lab code or the generated Excel sheet.

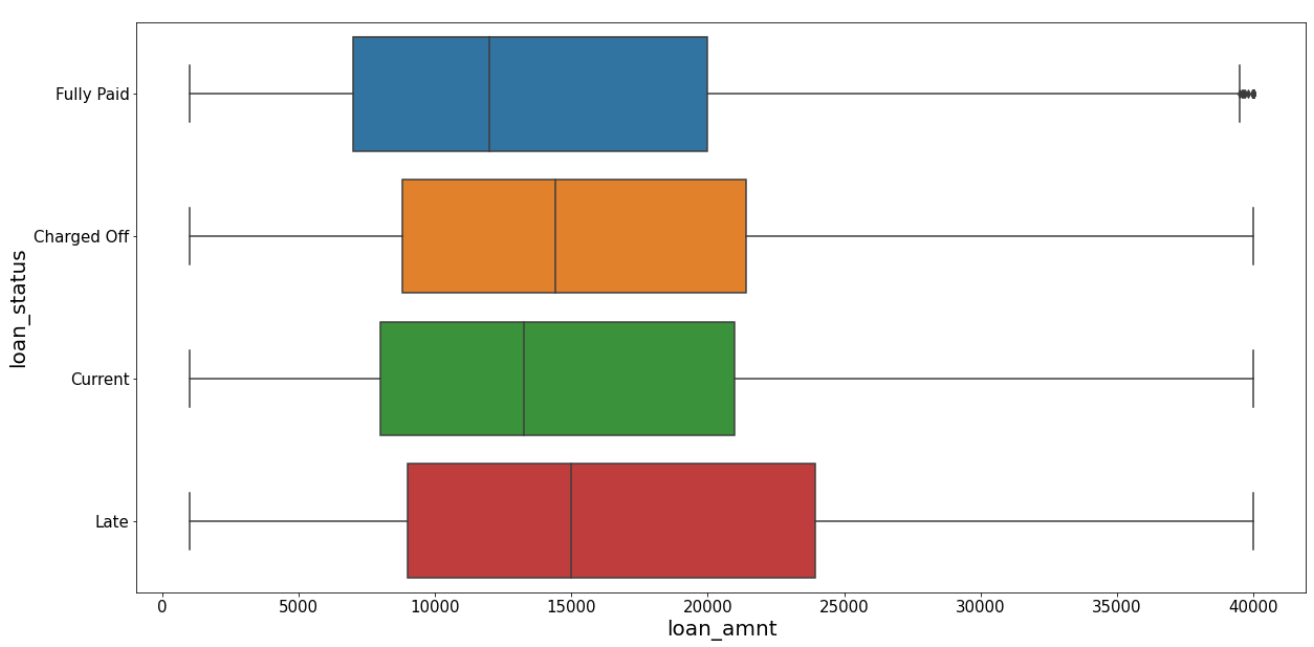
*Figure 1. Lending Club Loans data dictionary preview*

While there were no financial requirements to consider during the feasibility analysis, personnel were deemed to be capable, and have sufficient time and resources (including sufficient data), to complete the high level task. Database Familiarisation was carried out during this stage by creating summaries for each of the variables, including the target variable.

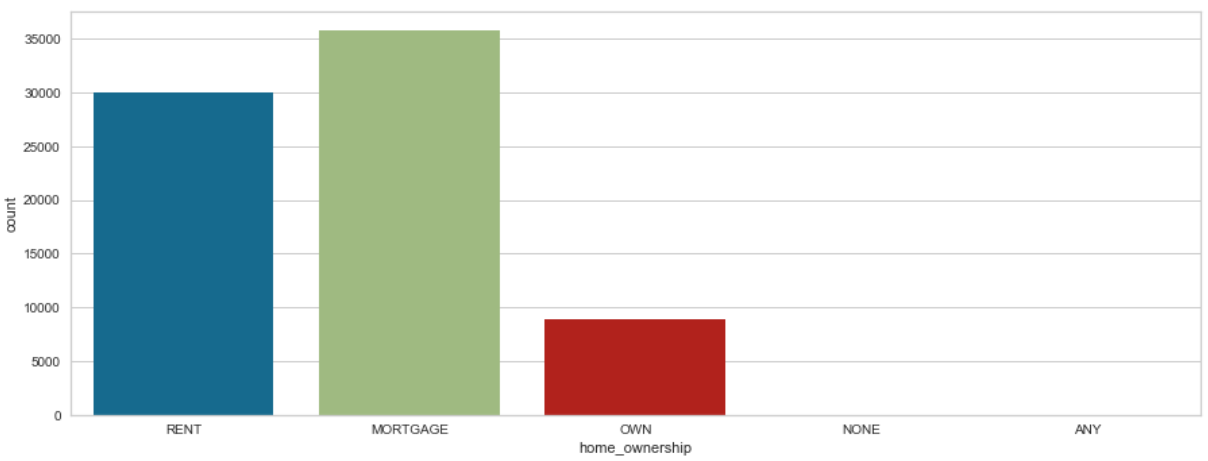
*Figure 2. Value counts for each of the target variable classes.*

At this point, the number of classes in the target variable was reduced from seven to four but this will be discussed in the data cleaning section. The data dictionary acted as a good summary for the number of missing and unique values for each variable, but it does not provide any information on the variation of data which is important when considering outliers of continuous variables.

*Figure 3. Indicator variable with many outliers.*

**To summarise this, box plots were created and these indicated which continuous variables had a large number of outliers and which had a small number of outliers as can be seen from figure 3 and 4 respectively.

*Figure 4. Indicator variable with a small number of outliers.*

Familiarisation with the categorical indicator variables was carried out in the same way as the indicator variable; by creating count plots. These count plots were useful in showing which categorical variables had under-represented classes and it allowed amalgamation decisions to be made which will be discussed in the data cleaning section.

*Figure 5. Categorical indicator variable with outliers.*

**2.2 Resourcing**

The resourcing stage involves gathering all the required resources for the project. This involves actions such as obtaining the database and transforming it into the operational database. The LCLDB was obtained by downloading it directly from the course portal on BlackBoard. All the data came pre-formatted, and all software and hardware requirements were already obtained to allow data-mining labs to be completed at home. Jupyter Lab was used on a HP Laptop with an i5 processor and 256GigaByte Solid State Drive.

**2.3 Data Cleaning**

The aim of the data cleaning stage is to prepare data for future processes that involve learning. Processes at this stage included sampling, balancing, missing data handling and erroneous outlier handling. This preparation of data should enhance the decision-making that follows the knowledge-discovery which is extremely important, especially in areas such as medicine (Lattar et al., 2020). Missing data handling was the first process to be carried out. As can be seen from the data dictionary, many indicator variables had a large amount of missing data. The majority of missing data was caused by a combination of accounts being opened for less than 6 months, individual accounts having no ‘joint account’ information (and vice versa) and most accounts having no hardship data due to no recorded hardships such as job loss etc.. To deal with the missing data from accounts opened less than 6 months ago, all the data from accounts that had been opened for less than 6 months was discarded, using the code from figure 7, which reduced the number of missing values across eleven different indicator variables to zero.

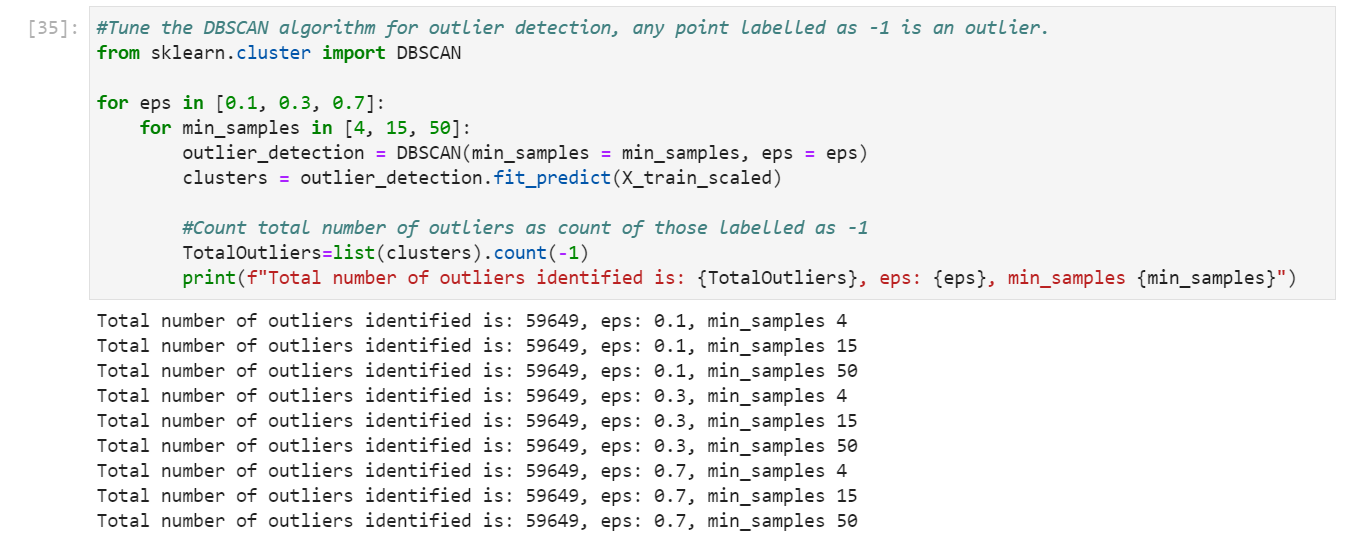


*Figure 7. Code to remove data from accounts opened within the last 6 months.*

To deal with the missing data due to hardship and joint accounts, alternative features were created and the original features were dropped. However, this will be discussed in the pre-processing section. All datetime variables were dropped during the data cleaning stage due to a lack of confidence to be able to use them appropriately within the learning stages. This did help to remove datetime variables with large amounts of missing data, such as ‘next\_pymnt\_d’ and ‘payment\_plan\_start\_date’.

*Figure 6. Datetime indicator variables description and removal.*

After the datetime data removal, there were still seventeen variables with missing data, fourteen of which had over 1000 missing values. The ‘emp\_title’ variable had 5704 missing values, but it was deemed too valuable to remove. At this stage, the ‘emp\_title’ variable has its missing values replaced by a class called ‘Unknown’ although, more steps were taken during pre-processing to enhance this variable further, as there were still many unique values. The ‘emp\_title’ variable was not the only variable that had issues with unique values, as another variable had every value as a unique value (‘id’) and a second variable had only 1 single unique value (‘pymnt\_plan’). Both of these variables were discarded as they would serve no benefit at the learning stage. Finally, the 13 remaining variables with over 1000 missing values were removed completely, although imputation was considered, and the 3 variables with less than 1000 missing values had entire rows removed. By handling missing data in this order and carrying out the pre-processing stage simultaneously, this meant a minimum amount of data was lost. The remaining processes of balancing, sampling and erroneous outlier handling were not carried out until the pre-processing stage had been completed which shall be further discussed here.

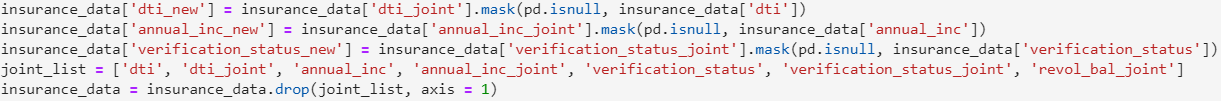
The first of the three processes to be carried out was sampling. This was done by separating the target variable of ‘loan\_status’ from all of the indicator variables and then passing them into the ‘train\_test\_split’ function in sklearn which created both training and testing data for the indicator variables (‘X’) and the target variable (‘y’). This split was stratified based on the values of the target variables which meant that the training and testing values for ‘y’ should have equal proportions of each class. Once this split had been carried out, erroneous outlier handling and balancing could be carried out. These processes were carried out only on the training data after sampling, as this means that no data should leak through into the test data. The erroneous outlier handling was carried out using an ‘Isolation Forest’ unsupervised learning algorithm on the ‘X’ training data which identified over 3000 outlying pieces of data, all of which were removed from both the ‘X’ and ‘y’ training data. The ‘Isolation Forest’ algorithm was chosen as it meant that data did not have to be pre-scaled and also due to a lack of success using the ‘DBScan’ algorithm, which regarded every single datapoint as an outlier, even when using scaled data.

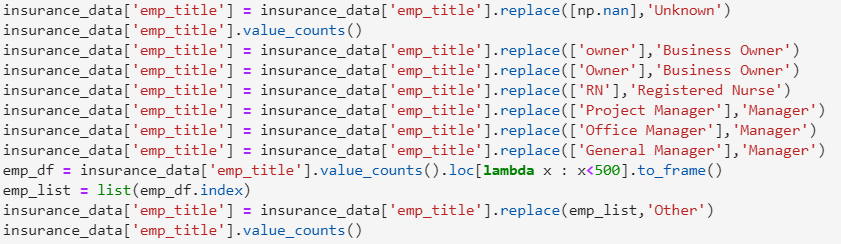
*Figure 7. Unsuccessful attempt at tuning DBSCAN algorithm.*

Once all these outliers had been removed, balancing of the training data could be carried out to ensure that each class within the target variable was equally represented during learning. This balancing process was done by combining the ‘X’ and ‘y’ training data and then up-sampling classes with less than 5000 occurrences and down-sampling data with more than 5000 occurrences, resulting in a training dataframe with 5000 occurrences of each class. Finally, the training dataframe was split to reform the ‘X’ and ‘y’ training dataframes.

**2.4 Pre-Processing**

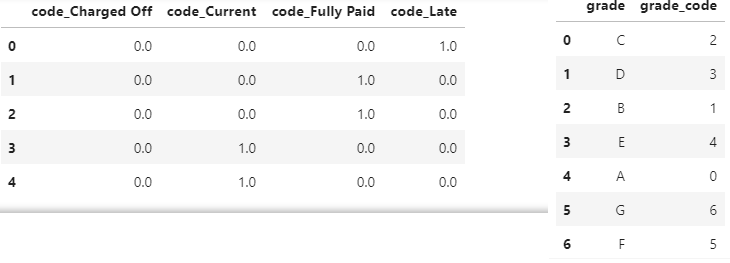
The pre-processing stage is extremely important to the KDD process and if carried out correctly and thoroughly, can yield knowledge that is sufficient to make decisions on. Along with the knowledge that can be gained at this stage, pre-processing will enhance the quality of the analysis at the learning stage (Svec et al., 2020). As already mentioned, the pre-processing stage was carried out simultaneously with the data cleaning stage. The 3 processes that were carried out at this stage were feature creation and feature selection and feature scaling. Feature creation was the first process that was actioned. A single feature for hardship was created which would indicate whether someone had spent a period in hardship or not, using ‘1’ and ‘0’ respectively. There was consideration at this stage whether the Boolean values of ‘True’ and ‘False’ should have been used but this option was decided against, due to uncertainty about Boolean behaviour within a model. Subsequently, all the other hardship related variables with missing data were discarded. Missing data caused by individual accounts not having any data for the ‘joint’ variables was handled by creating single variables for annual income, verification status and debt-to-income which was equal to the joint variable data if it was present and was equal to the individual variable data if it wasn’t present. The ‘revol\_bal\_joint’ variable was discarded without any corresponding feature being created as no corresponding individual account variable could be identified.

*Figure 8. Code to replace individual/joint account data with a single variable for each.*

**Following the removal of the joint account data a focus was put onto the variables with an ‘object’ data type (categorical variables). Some of the object variables had many underrepresented classes, some of which could possibly have been mistakes, as was seen in figure 5, where ‘NONE’ and ‘ANY’ classes had only one occurrence. This was resolved by replacing the ‘NONE’ and ‘ANY’ classes with the modal classes of ‘RENT’ and ‘MORTGAGE’ respectively, thus, reducing the total number of classes within this variable to 3. Further class reduction was carried out on the ‘purpose’ and ‘emp\_title’ variables which classified the reason for the loan and the clients job title respectively. This class reduction was carried out as database examination exposed multiple classes which should fall under the same class and also because there were many underrepresented classes that needed to be amalgamated under a larger umbrella class in order to be useful at the learning stage and not bias the learning towards a decision tree or random forest based model, which is very good at handling multi-class categorical variables. The ‘purpose’ variable was reduced from 12 to 9 by amalgamating the ‘major\_purchase’ and ‘renewable\_energy’ classes and by creating a new class called ‘home’ which acted as an umbrella for the ‘house’, ‘moving’ and ‘home\_improvement’ variables. Figure 9 shows how class reduction was carried out on the employment title variable.

*Figure 9. Class reduction across the ‘emp\_title’ variable.*

After the processes of variable combination and class reduction were carried out, feature construction was completed by encoding the classes to allow mathematical based algorithms to handle the data. This was done using the sklearn ‘LabelEncoder’ for all of the indicator variables while the target variable was encoded using ‘OneHotEncoder’. The former returns a single variable with each class assigned a different integer number within the range 0 : (No. of classes – 1) while the latter returns multiple variables (one for each class) with a 1 placed within the associated variable and a 0 placed within the remaining unassociated variables. Examples of each can be found in figure 10.

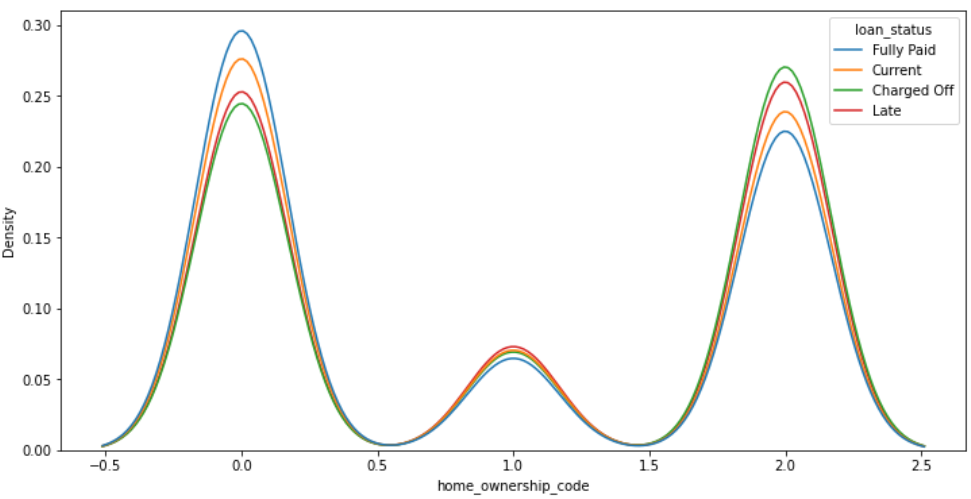


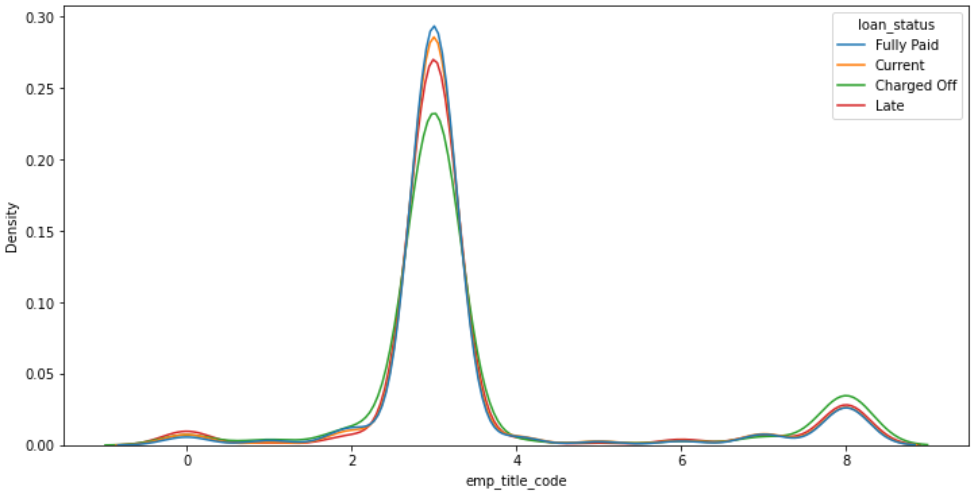
*Figure 10. Example of ‘LabelEncoder’ and ‘OneHotEncoder’*

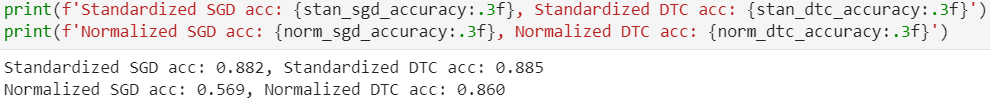
Following the encoding, unencoded indicator variables were removed from the dataframe. However, the opposite was done for the target variable as the encoded columns for loan status were only required for the unsupervised learning model evaluation which will be discussed later. Before the encoded columns were removed, an instance of them was created for future use.

The next process within the pre-processing stage was feature selection, which aims to reduce the number of features, or indicator variables, within the model. This was done in part by removing variables with large amounts of missing data during cleaning and by combining variables at the beginning of pre-processing. This meant that the number of indicator variables had already been reduced from 107 to 75. To reduce this further, highly correlated indicator variables were identified using the corr() function. Once correlations between pairs were calculated, they were placed into a dataframe alongside the absolute value of their correlation. A filter was applied to this dataframe so remove all permutations with a correlation coefficient of less than 0.8 and also any permutations equal to 1. The drop\_duplicates() function was then passed due to each correlation having an inverse correlation which was exactly the same absolute value and this meant that the resulting dataframe only contained a single permutation for each correlation. Having a single permutation of each correlation in the dataframe meant a list could be created based on the dataframe index, and this list could be used to drop variables from the original dataframe for analysis. Once these highly correlated indicator variables had been removed, there were fifty-two indicator variables remaining. From these fifty-two remaining indicator variables, thirty were selected using the sklearn ‘SelectKBest’ method along with mutual information acting as the scoring function parameter. This resulted in a list being produced that contained the thirty indicator variables with the highest mutual information score when compared to the target variable. The mutual information score allows us to measure the relevance of each indicator variable to the target variable (Ircio et al., 2020).

The penultimate step of the pre-processing stage involved visualising each of the indicator variables using a kernel density estimate (KDE) plot with the aim of discovering a small amount of knowledge going into the learning stage. Three interesting pieces of information resulted from this, one of them being; there was a higher prevalence of late and charged off loans amongst loanees that rented their homes, and a lower prevalence amongst loanees that were paying a mortgage. The prevalence of each loan classification for loanees that owned their home was even, so it could not be said that owning a home would make you more or less likely to pay back a loan, which can be seen from figure 11. Secondly, figure 12 then shows another interesting piece of information which is that loanees with a relatively uncommon job (one which had been put under the umbrella of the ‘Other’ class) had a lower prevalence of late and charged off loans, while loanees with an ‘Unknown’ job type had a higher prevalence. Finally, it was also discovered that there was a higher prevalence of late and charged off loans amongst loanees who needed the money for the purpose of debt consolidation, while there was a lower prevalence amongst loanees who needed the money to pay off their credit card bills.

*Figure 11. Home ownership code KDE plot with ‘loan\_status’ acting as ‘hue’ parameter.*

*Figure 12. Employment title code KDE plot with ‘loan\_status’ acting as ‘hue’ parameter.*

Feature scaling, the final step in the pre-processing stage could be handled very quickly by the hardware using the sklearn ‘StandardScaler’ method. Normalisation was also considered at this stage but after comparing model performance of both a random forest pipeline and a stochastic gradient descent pipeline, the standardised option was chosen as it produced a higher accuracy as shown in figure 13.

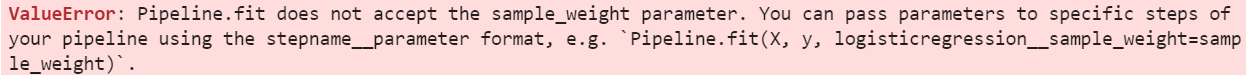
*Figure 13. Normalisation and standardisation feature scaling options.*

Standardising the data, results in each of the features having the properties of a standard normal distribution with a standard deviation equal to one and a mean equal to zero. This is important so features containing large numbers do not bias the model. Principle component analysis was also considered at this final stage, but it wasn’t deemed necessary as hardware had sufficient processing power to handle the training data in its entirety.

**2.5 Supervised Data Mining**

Unlike previous stages where each portion of code could be handled almost instantaneously by the processing hardware, the data mining stage required some patience to allow the learning algorithms to iterate across the data many times. A high level task requirement was to show how both a supervised and an unsupervised learning algorithm performed when fitted to the training data. The difference between supervised and unsupervised learning is that during supervised learning, the algorithm is allowed to see the training values for the indicator variable, which informed it whether the decision it has come to is right or wrong. During unsupervised learning, the indicator variable is hidden from the algorithm and it makes its own decision on what cluster each data point belongs to. Model selection for the supervised classification was carried out first. To select the best model, it was necessary to compare the accuracies of multiple algorithms. The decision was made to fit the training data to five different classifier models, predict the classes of the testing data and then produce a classification report and confusion matrix for each. The three best of these five models would then go forward to form an ensemble classifier which would go through a ‘bagging’ process and be compared with a random forest ensemble classifier.

Initially, the five original classifiers were trained using a pipeline which had a standardisation process followed by a classification process. However, this was unsuccessful in the later stages when the ensemble of pipelines was passed into a bagging classifier as it produced an error which was not understood despite researching the problem.

*Figure 14. Error occurrence during Bagging.*

To prevent this problem from occurring when this stage was reached, the decision was made to pre-standardise the data by creating instances of standardised dataframes which could be used in each of the classifiers. To prevent any bias created by the ‘StandardScaler’ algorithm leaking through into the testing data, the algorithm was fitted only to the testing data, but used to transform both the testing and trained data. The pre-standardised data was then passed directly into each classifier instead of via a pipeline.

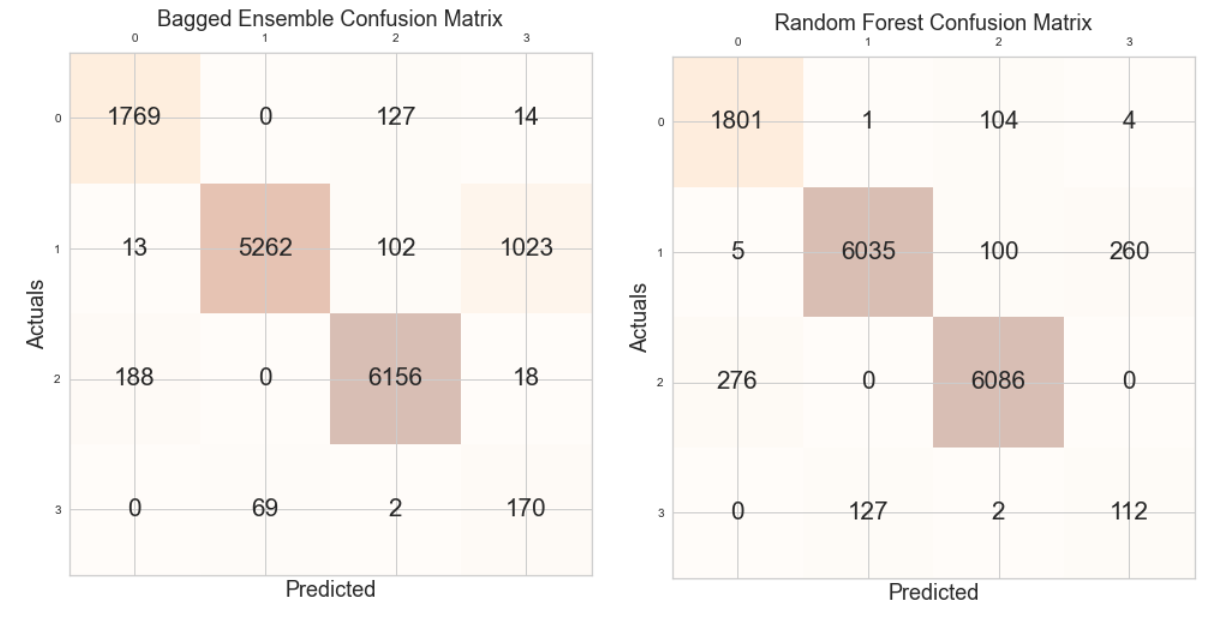
**The preliminary evaluation of the models resulted in selecting logistic regression, stochastic gradient descent and gaussian naïve bayes classifiers to go forward into a voting classifier ensemble which would be compared to a random forest and gradient boost ensemble.

*Figure 15. Preliminary evaluation results for five classifier algorithms.*

As this was a multi-class classification (more than two classes), the precision, recall and F1 score in figure 16 were calculated as a macro average and not as a weighted average. The macro average was opted for as this created a bias towards the classes of ‘Late’ and ‘Charged Off’ which had a lower representation. This bias was felt to be beneficial as it could be argued that it is more important for a lender to know if a loanee is going to be unable to pay the loan back rather than if they are going to be able to pay it all back on time. This also explains why it wasn’t the accuracy score that was used to select the models , as the accuracy score is weighted towards the representation of each class. Due to the random nature of the stratification process that took place during the test/train split, each time the code re-ran from the beginning, each of the values in figure 16 changed very slightly. The models were not selected based on F1 score as this assumes that there is an equal importance placed on both precision and recall, but for the lender, recall will be even more important as it will allow them to maximise the percentage of ‘bad’ loanees they identify.

**Prior to the preliminary evaluation, each of the models were hypertuned to ensure that they were performing as well as they could. Due to hardware processing limitations, not as many parameters could be tested as was desired. Figure 17 shows an example of how the logistic regression algorithm was hypertuned by grid searching.

*Figure 16. Logistic regression hypertuning.*

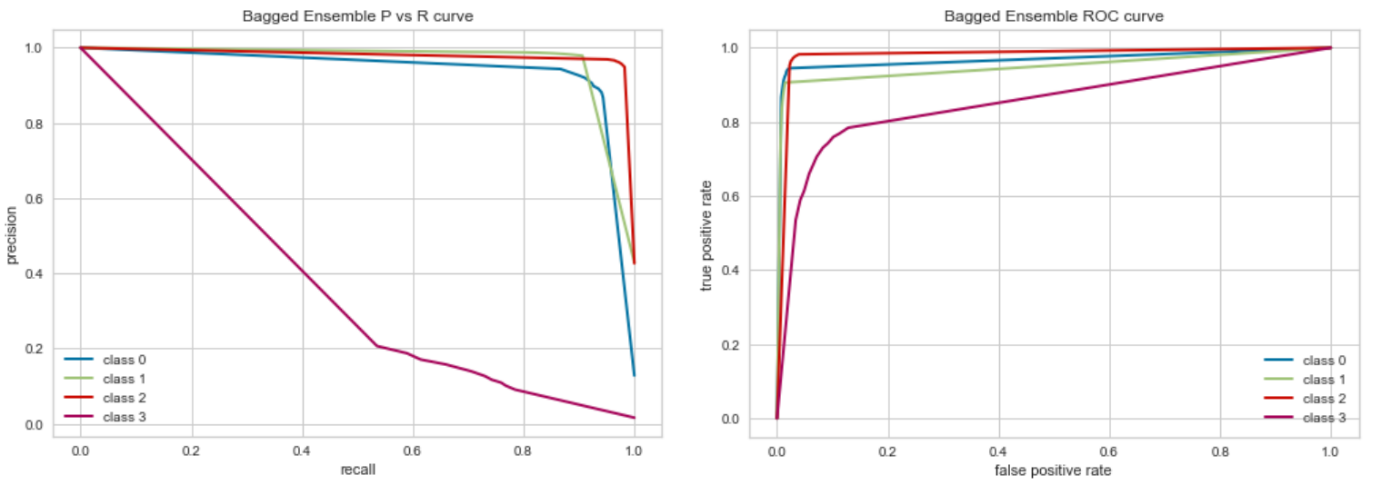
After the preliminary evaluation of the hypertuned models, the three selected models were passed into a voting classifier to create an ensemble model, which aimed to utilise the specific accuracies of each model. This was then taken a step further by passing the custom ensemble into a bagging (bootstrap aggregation) algorithm which aims to reduce the variance in a classifier by creating several random subsets of data and training the classifier on each of these subsets. This bagging process is like the process carried out by the random forest algorithm which goes a step further and selects random features of random subsets to train decision tree classifiers. The preliminary evaluation of the two ensemble classifiers has been summarised in figure 18 and the confusion matrices are shown below in figure 17. The random forest had an accuracy of 0.94 while the bagged voting classifier had a lower accuracy of 0.9. However, as mentioned previously, the recall value for the ‘Charged Off’ and ‘Late’ classes will be most important to the lender. For this reason, the bagged ensemble classifier was selected as the most appropriate model for a lender to classify loanees.

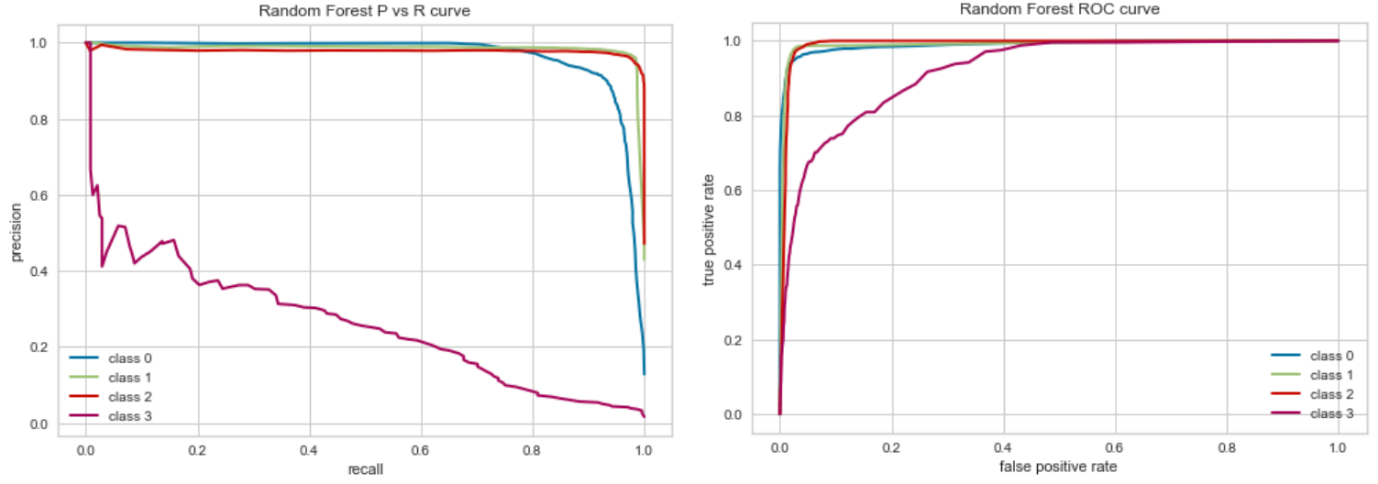
*Figure 17. Confusion matrices for each ensemble.*

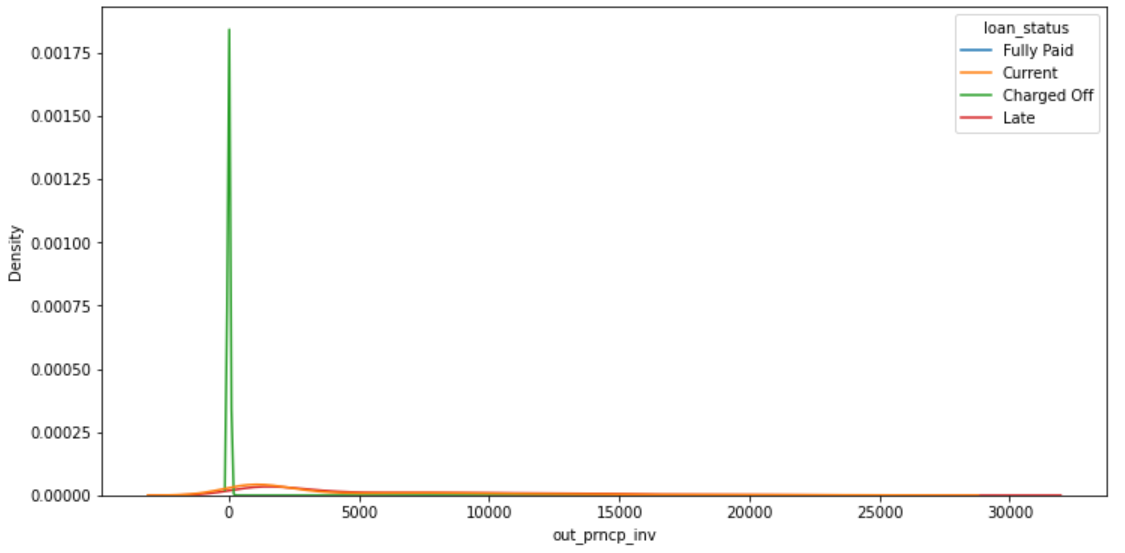
*Figure 18. Preliminary evaluation of ensembles*

**2.6 Supervised Evaluation**

The evaluation stage aims to assess the model’s suitability and its flaws. This stage has blurred boundaries with the data mining stage, as was seen by the preliminary evaluation that was carried out. Both ensemble methods went a step further than the preliminary evaluation by having precision vs recall curves and ROC curves plotted. These graphs help to visualise the performance of the model. These curves were plotted using the encoded data for the target variable from the pre-processing stage which allowed the precision and recall for each class to be visualised as it was a multi-class classification. A macro averaged precision vs recall and ROC curve had been planned, but due to time limitations this was not achieved.

*Figure 19. Precision vs recall curves for bagged voting classifer ensemble.*

*Figure 20. Precision vs recall curves for random forest ensemble.*

Both sets of precision and recall curves follow a similar pattern with the models performing extremely well for class 0, class 1 and class 2 which correspond to the classes of ‘Charged Off’, ‘Current’ and ‘Fully Paid’ respectively. Class 3, ‘Late’, did not perform as well. The ‘Late’ and ‘Charged Off’ classes were regarded as the most important as these are the loans that will cost the lender money, but the models ability to identify each class varied greatly. This is due to a decision that was made during feature, which was to retain the ‘out\_prncp\_inv’ feature which contained data on the **‘remaining outstanding principal for portion of total amount funded by investors’.

*Figure 21. KDE plot showing distribution of ‘out\_prncp\_inv’ values for each class.*

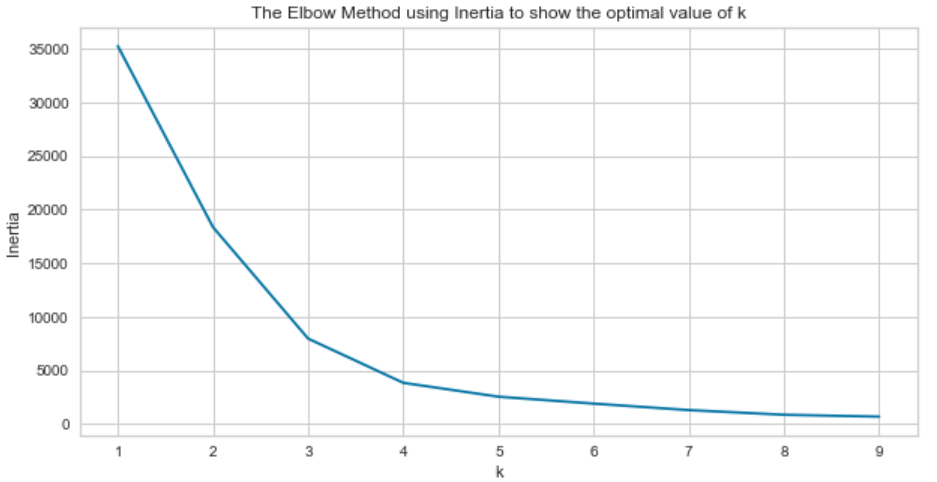
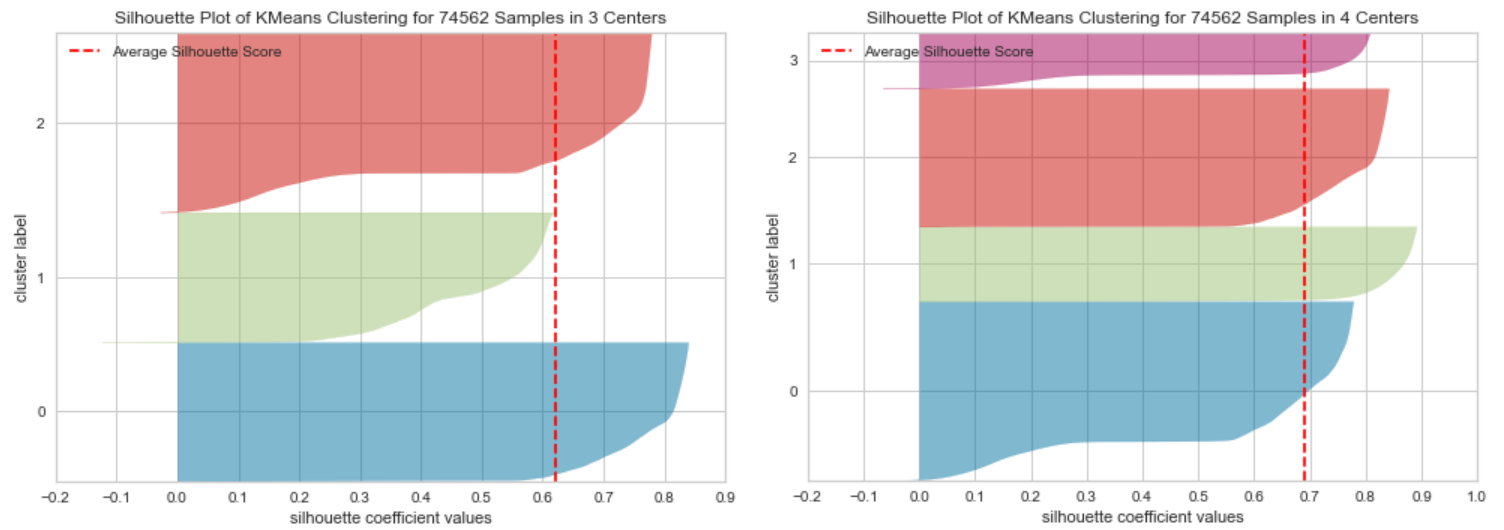
As can be seen from figure 21, every loan that had been charged off contained a zero value for this feature. This meant that learning algorithms could use this single feature to accurately predict if a loan should be classified as ‘Charged Off’. At the pre-processing stage, this feature should have perhaps been removed.

**3.0 Unsupervised Learning**

**3.1 Unsupervised Data Mining**

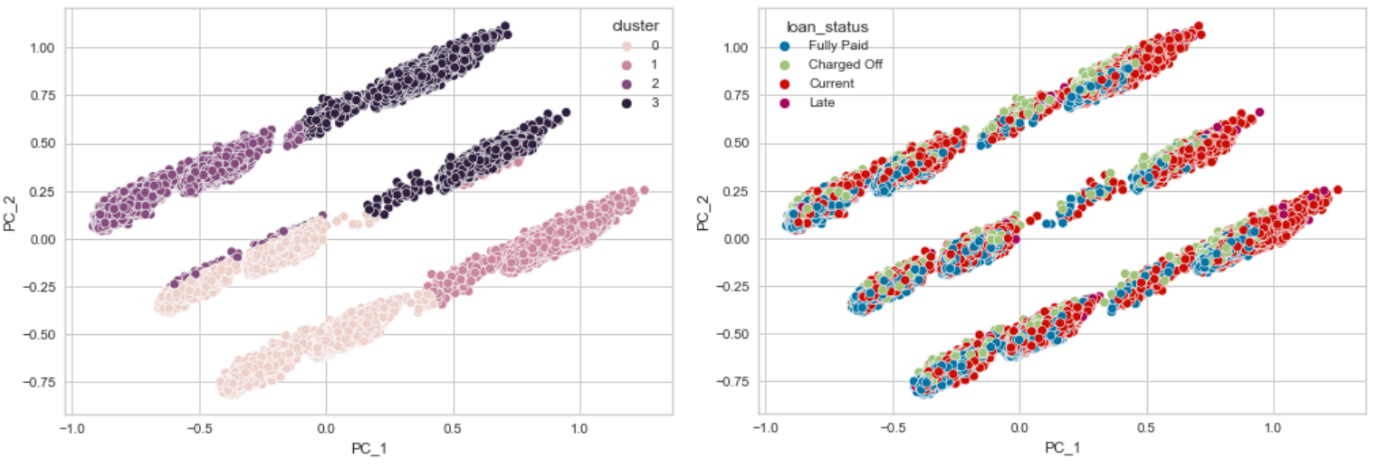
Unsupervised learning does not provide the model with a target indicator. Instead, the algorithm forms clusters of similar data. During the unsupervised learning, an alternative method was used to scale the data which was called a MinMax Scaler. This scaler ensured that each value within a feature was scaled to within a minimum and maximum value which were set at 0 and 1 respectively by default. Additionally, principle component analysis using the PCA method allowed the total number of features to be reduced to two, which allowed for easier two dimensional visualisation of the clusters in the evaluation stage.

The most important parameter for the K-Means clustering algorithm is the number of clusters that you would like it to form. For this project, since the total number of target classes was already known, this was the most logical choice of clusters but, in many cases, there will not be a known target class. To evaluate what the optimum number of clusters may be, two visualisations were created. The first, and quickest, visualisation displayed the elbow method. The elbow method using inertia was opted for, which is the sum of squared distances of samples to their closest cluster centroid. The results from the elbow method graph indicates that the optimum number of clusters as either three or four, as beyond this number, there is very little decline in inertia. Despite this information, four clusters were opted for, as it was already known that it was a four-class problem. The second visualisation indicated what the average silhouette score is, which is a measure of the similarity of points within a cluster and their distance to points of a different cluster. The higher this value, the better defined each cluster will be.

*Figure 22. The elbow method using inertia.*

*Figure 23. Silhouette score for 3 clusters and 4 clusters using Kmeans algorithm.*

**3.2 Unsupervised Evaluation**



*Figure 24. Unsupervised learning cluster formation of two principal components*

Despite a promising result from the silhouette score and low level of inertia, the unsupervised learning algorithm did not cluster the data based on the hidden loan status feature. With four clusters having a silhouette score of 0.68 compared to three clusters at 0.63, it was expected that the 2 principal components would form four very distinctive clusters. However, as can be seen from figure 24, there were in fact 3 very distinct clusters created but none of the cluster corresponded to the location of the loan status classes. This lack of distinction between loan classes was reinforced by plotting the number of each loan status classes that fell under each cluster, which seemed almost completely random. Due to space limitations this count plot cannot be shown in the report, but can be seen within the submitted code.

**4.0 Conclusion**

The supervised learning algorithm performed extremely well at identifying both good loanees and bad loanees. However, as mentioned at the end of section 2.6, the presence of the ‘out\_princ\_inv’ feature is likely to have made it very easy for the algorithm to identify loans that had been charged off. While the random forest algorithm produced a better accuracy, the voting classifier ensemble was better at identifying ‘bad’ loanees, which would be more valuable to the lender. Future models could improve their recall for bad loans by using the ‘class\_weight’ parameter to bias the model towards the relevant classes. However, to significantly improve the recall for the ‘bad’ loanees, this would lead to many ‘good’ loanees being misclassified. An excessive misclassification of good loanees could result in lost income beyond what the ‘bad’ loanees are costing. The unsupervised learning algorithm was not as successful at identifying loans by their class. While the algorithm clearly created three distinct clusters from the data, none of these clusters seemed to represent a loan status. For these reasons, the supervised learning algorithm will be more valuable to the lender, but more work should be done to measure model performance using only features that can be measured prior to making a loan agreement.

**5.0 Bibliography**

AHIMA e-HIM Workgroup on EHR Data Content. (2006) 'Guidelines for developing a data dictionary' Journal of AHIMA, 77(2), pp. 64A-64D.

Choi, W., Kim, J., Lee, S. and Park E. (2021) 'Smart home and internet of things: A bibliometric study', Journal of Cleaner Production, 301, 126908.

Ircio, J., Lojo, A., Mori, U. and Lozano, J. A. (2020) 'Mutual information based feature subset selection in multivariate time series classification', Pattern Recognition, 108(1), pp. 107525.

Lattar, H., Salem, A. B. and Ghezala, H. H. B. (2020) 'Does data cleaning improve heart disease prediction?', Procedia of Computer Science, 176(1), pp. 1131-1140.

McCabe, A., Nic An Fhailí, S., O'Sullivan, R., Brenner, M., Gannon, B., Ryan, J., Butt, A., Schull, M. and Wakai, A. (2019) 'Development and validation of a data dictionary for a feasibility analysis of emergency department key performance indicators', International Journal of Medical Informatics, 126(1), pp. 59-64.

Redman, T. C. (2016) Bad Data Costs the U.S. $3 Trillion Per Year, Available at: https://hbr.org/2016/09/bad-data-costs-the-u-s-3-trillion-per-year (Accessed: 11th May).

Svec, P., Benko, L., Kadlecik., M., Kratochvil, J. and Munk, M. (2020) 'Web Usage Mining: Data Pre-processing Impact on Found Knowledge in Predictive Modelling', Procedia of Computer Science, 171(1), pp. 168-178.

Venayagamoorthy, G. K. (2014) 'Computational Approaches for Bad Data Handling in Power System Synchrophasor Networks', IFAC Proceedings Volumes, 47(3), pp. 11269-11274.