Student: 2721301

CSCU9T6 Assignment 2021

Contents

[Business understanding 2](#_Toc69051386)

[Data understanding 3](#_Toc69051387)

[Data Preparation 5](#_Toc69051388)

[Modelling 6](#_Toc69051389)

[Results and Errors 12](#_Toc69051390)

# Business understanding

I received a task from a bank client which desires to have a system that can predict whether a bank customer is likely to repay a loan taken out or default it. The task will require a classifier model which will learn from data received from the bank on previous clients and the loan outcome, after learning from this it will be able to accurately predict the outcome of a new client based on the data they provide.

This task is suitable for a data mining approach as data mining is a very powerful tool that can find patterns in raw data that would be difficult or near impossible for humans to notice if there is a pattern in the first place. It can turn this raw data, called variables, into suitable information using a model, which will help companies better understand their customers, and allows the company to work more effectively by minimising loss. For this situation it will prevent the bank losing as much money by only granting those that the system predicts will pay it back, depending on previous data. This system can predict outcomes that analysis techniques from the past are just not capable of making, making it a very powerful tool for the future of businesses.

As previously mentioned, the system uses a model which is a way of transforming data to produce an output. It is a set of variables that are interpreted by software using one of many data mining techniques, the technique depends on the task received. For example, a model can estimate how long it would take to boil a kettle by using previous data on that kettle. The data can be called a variable, in this case, it could be volume of water added.

A variable is a data point and is something we measure which can vary. The first step of any data mining task is identifying the variables, which can be done by viewing the data provided. A variable can take numerous different forms, from discrete or continuous numerical data, to nominal or ordinal categorical data.

The methodology I will be using to create this model is the CRISP-DM framework and the following is a short summary of the stages.

* **Business understanding:** Try to understand the task, identify what we are trying to find out and what type of task it is (classification, regression, clustering)
* **Data understanding:** Try to understand the data provided by determining what the data is on, look at what variables there are and identify which are relevant to the task, what the target variable is and possibly plot some basic statistics on it to try better understanding it.
* **Data preparation:** At this stage we clean and format the data, we check for missing values, obvious errors, minority values, unbalanced data, skewed distribution and any inconsistencies, an inconsistency, for example, could be saying ‘m’ instead of ‘male’. We determine which variables will be included in our model by looking at the noise and distributions of them, and then split the data into a training set, test set and sometimes validation set.
* **Modelling:** We choose a technique to use which is called a model, and then train the training data to this model. We can tune this model and compare its outputs until we are happy with it and can use it on the test data.
* **Evaluation:** We use the model on the unseen test data and measure how well it works, if we are satisfied with the results, we can move on to the deployment stage.
* **Deployment:** Our chosen model is implemented into a system to be used; the model is continually reviewed as the nature of it may change over time.

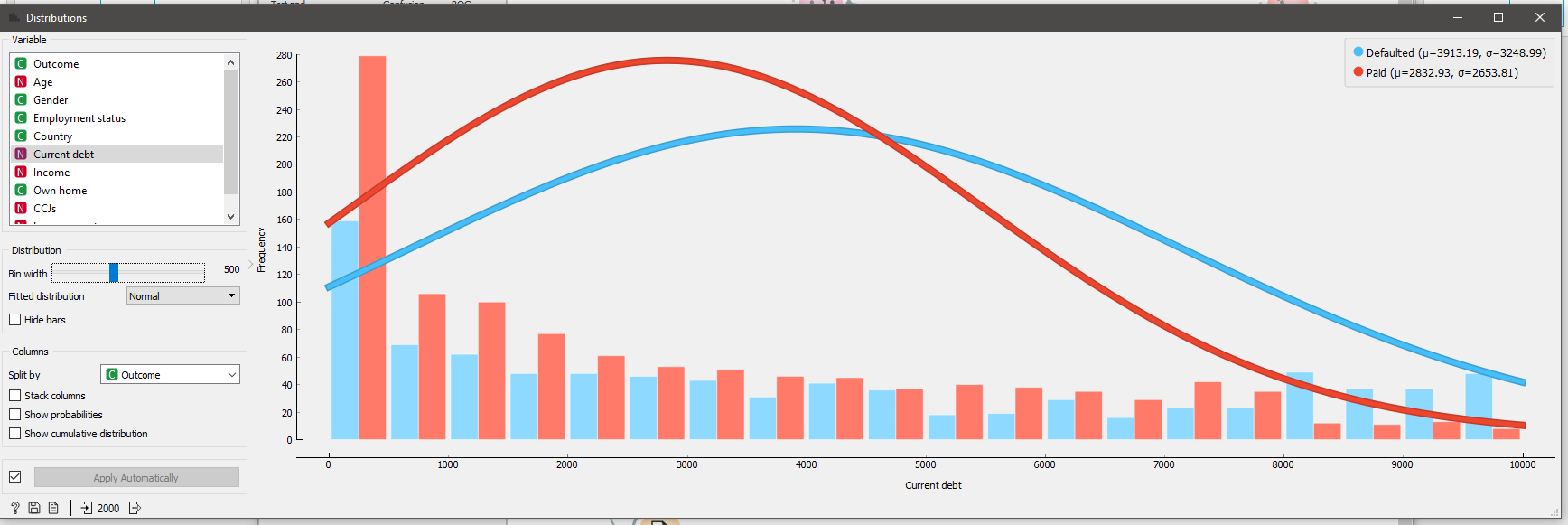
With CRISP-DM we can go back and forth between stages as we learn more about the data and task.

# Data understanding

The following is a table consisting of the variables I found within the data I provided. I will display what type of variable it is, what role it has in the system, and justification of why I gave it that role.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Role** |
| Fictional Surname | Text | Not used |
| Postcode | Text | Not used |
| CustomerID | Text | Not used |
| Age | Numerical Discrete | Input |
| Years at address | Numerical Continuous | Input |
| Employment status | Categorical Nominal | Input |
| Country | Categorical Nominal | Not used |
| Current debt | Numerical Continuous | Input |
| Income | Numerical Continuous | Not used |
| Own Home | Categorical Nominal | Input |
| CCJ’s | Numerical Continuous | Input |
| Loan amount | Numerical Continuous | Input |
| Outcome | Categorical Nominal | Target Variable (Output) |
|  |  |  |

* **Functional Surname, Postcode, CustomerID**: These are not used as they are meaningless for this task and will not provide us with any information.
* **Age:** This is included as there is a serious variance on whether they will pay back depending on age. For example, they are a 61.7% likely to pay back if in the 65-70 age group, compared to 49.4% with the under 20 age group.
* **Years at address:** I decided to remove this after looking at histograms split by the outcome as it shows that it does not influence the outcome, all bins for years show 50-55% paid and 40-45% defaulted and is consistent.
* **Employment status:** I included this as not being employed can reduce the chance to pay back the loan as the customer likely does not have a stable income, although the histograms show a minimal change between them, employed are 60% likely to pay back and 55% for employed and unemployed. I will talk about why I have not considered retired in the data preparation stage.
* **Country:** I have decided not to include this as a large majority of the values are UK and the rest have a very small amount are other countries, I will discuss this more in data preparation.
* **Current debt:** This is included as it heavily effects the chance of the loan being repaid. As shown in the distribution chart below with a normal fitted distribution line added, we can see that as current debt increases, so does the chance of the customer default the loan.



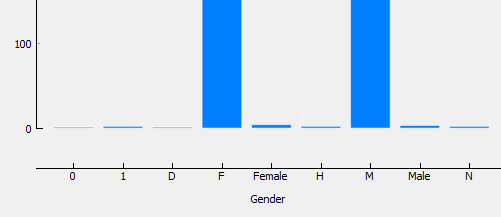
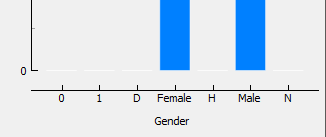
* **Income:** I initially chose to add this variable as I assumed the higher amount earned, the more likely the customer is to pay. However, after looking at the distributions I can see that every group between £10,000 and £55,000 are around 42-46% chance likely to pay back the loan, so I have decided not to include this.
* **Own home:** This is included as the histogram shows that those who own their house are more likely to pay with a 59.9% chance, whereas renting has 54.5% and mortgaging has 51.1%, showing a clear relation with this and the outcome.
* **CCJ’s:** This variable is a county court judgement where a person is called to pay money they owe to a debt, with this we can assume the more CCJ’s a person has then the less likely they are to pay. Thich is backed up with the histogram showing the chance of paying drastically drops for those with 2 or more CCJ’s.
* **Loan Amount:** The histogram does not show a relation between this and the paid back outcome so I will not add it.
* **Outcome:** This is the target variable we are trying to predict; it is the output on whether a customer will pay back their loan.

# Data Preparation

The first action to take at this stage is to split the data into training and test data, so that I will only be using the training data to tune the model. The test data will remain unseen until I am happy with my model and decide to test it on it. Using a data sampler, I split the data 70% for training and 30% for test.

I cleaned the data, making the following changes:

* The data for gender is mostly ‘F’ for female and ‘M’ for male, however there are some values for ‘male’ and ‘female’ present. I merged the F to female and the M to male, I got rid of the other values ‘0’, ‘1’, ‘D’, ‘H’, ‘N’ as they do not represent a gender. The images below show the histograms before and after making this change, using an “edit domain” tool.



* I decided not to include data from countries that are not the UK. I did this as other countries such as France, Germany and Spain are minority values, making up a combined total of 0.3% in the data.
* For employment status I decide to remove people from the ‘retired’ category as they are minority values, making up 0.25% of the overall data.

I then scaled the data making the following changes:

* I decided not to include those with over 3 CCJ’s as there are only 2 and are large outliers, 10 and 100, which would heavily affect the data.
* For income I saw that those making below £10,000 are minority values, and there are 2 outliers of those making over £55,000. Due to this I have decided to only consider those earning between £10,000 and £55,000

# Modelling

**Validation method**

A problem that can occur when modelling is tuning our model too much to the training data, meaning it may not do well to adapt to different data. When we split up the data into training set and test set, we can be unlucky with the split, and mild outliers or hard to predict data can fall into the test set. This means that our model will not be able to learn from this kind of data and the testing will not perform as well as the training data. Due to this, we cannot model using the simple approach and must use a validation method.

The validation method I will be using for my modelling is K-Fold method. This method means we split up the data into many training and validation sets to tune the model with. With this we can see what the average performance of our model is across these different sets, and how it can deal with completely random smaller sets of data. Doing this it allows our model to cope better with potentially much different data between the sets.

For my model I will be using 10 different folds of data, each with a training set and validation set. With this we can find the average error across these 10 folds, which allows us to see the variance across the model and stops us relying on 1 set of data.

**Modelling methods**

When modelling we need to find a way to have a balance between bias and variance, or underfitting and overfitting. The three models I will be trying for this classification task are Decision Tree, K-Nearest neighbour, and random forest. At this stage I will only be using the training data and validation data, and I will leave the test data for the end when I choose which model to use. Below is a brief description of these methods.

* **Decision Tree:** This method is commonly used for classification tasks. It works by progressively splitting our data into smaller sets with the goal of dividing it to a point where it can be classified, with a label. A decision tree has a tree layout, and we are trying to divide our data as much as possible with this tree. We can adjust the hyperparameters such as how many splits we want, which can give us better accuracy. The number of splits we specify is known as the specifying the trees depth. If we make the tree depth too small, we will not get as many divisions and will have an underfitted tree, however if there are too many divisions it will become overfitted and wont generalise as well. An advantage of this method is that it is very easy to interpret by looking at the tree, however it is normally outperformed by more complex models. This method can be unstable as making small adjustments or adding new data can change the tree drastically.
* **K-Nearest Neighbour:** This method, also known as KNN, is an instance-based method. This means that it does not build a model, but instead it classifies a row of data by measuring how far away it is from the nearest labelled data point that the model has seen before. Each point of data is represented by a row of our training data and can be shown on something like a scatterplot. On this plot we can see how close it is to nearest classified data points, hence the name nearest neighbour. For a new data point it will predict its label based on its nearest neighbour. This method is very prone to outliers so we can adjust the number of neighbours required before it labels a new data point, this is a part of this models hyperparameters. If a new data point is an outlier, we can then look at its, for example, 5 nearest neighbours instead of just 1 to allow the model to make more accurate classifications. Choosing the number of neighbours is very important as if we specify too few, the classifier will generalise poorly and overfit, too many and we get an underfitted model which only work with the label that has the most instances in the scatterplot. The advantages of this are that it is easy to implement and does not require a complex model meaning the training time for this is very small. However, it must keep all the training data in memory so it can find the nearest neighbour, this results in very high memory usage and can lead to very slow predictions if the amount of data is high. This method is also not very good at dealing with outliers.
* **Random Forest:** This method is very similar to a decision tree; however, it is instead many decision trees that work together to make a classification. Each tree in the random forest produces a predicted class label for the data point, it counts the number of times this point has been labelled this way and the label with the highest majority wins, becoming the model’s prediction. There are many trees that work unrelated to each other that work together, having a committee making multiple classifications can outperform other single prediction models, making it more accurate. The trees protect each other from individual errors of that tree, meaning that one bad tree will not affect the overall outcome, as where 1 tree is wrong, the rest could be correct. The hyperparameters for this allow us to specify the number of trees we want to use and can limit how deep these trees go. Like decision trees, too few trees and depth will result in and under fitted model, and too many trees with larger depth can produce an overfitted model. This model, however, requires large computational power to build these trees and may take a lot of training time.

**Search method (Hyperparameter search)**

When modelling we get the cost function from the validation metrics to see how well our model performed. Our goal when modelling is to find the sweet spot with the greatest value for the cost function, which in this case will be between 0 and 1. We adjust the model parameters known as hyper parameters each time we run a test until we cannot get a better cost function score.

The method I will be using to adjust the hyperparameters of my models is the grid search method. Grid search is used to carry out hyperparameter optimisation. It is a method to find the optimum values of the hyper parameters for our model to give the best cost function value. My grid search table showing all my hyperparameters and output prediction values can be seen below.

**Grid search results**

Decision tree

Hyperparameters: {min number of instances in leaves, limit tree depth}

Results: {AUC, CA, F1, Precision, Recall} – The higher number the better

|  |  |
| --- | --- |
| **Hyperparameters** | **Results** |
| {5, 2} |  |
| {5, 5} |  |
| {5, 10} |  |
| {5, 15} |  |
| {5, 20} |  |
| {5, 25} |  |
| {5, 30} |  |
| {4, 20} |  |
| {5, 20} |  |
| {6, 20} |  |
| {7, 20} |  |
| {8, 20} |  |
| {9, 20} |  |
| {10, 20} |  |

K-Nearest Neighbour

Hyperparameters: {Number of neighbours, Metric}

Results: {AUC, CA, F1, Precision, Recall} – The higher number the better

|  |  |
| --- | --- |
| **Hyperparameters** | **Results** |
| {2, Manhattan} |  |
| {4, Manhattan} |  |
| {6, Manhattan} |  |
| {8, Manhattan} |  |
| {10, Manhattan} |  |
| {12, Manhattan} |  |
| {14, Manhattan} |  |
| {2, Euclidean} |  |
| {4, Euclidean} |  |
| {6, Euclidean} |  |
| {8, Euclidean} |  |
| {10, Euclidean} |  |
| {12, Euclidean} |  |
| {14, Euclidean} |  |

Random Forest

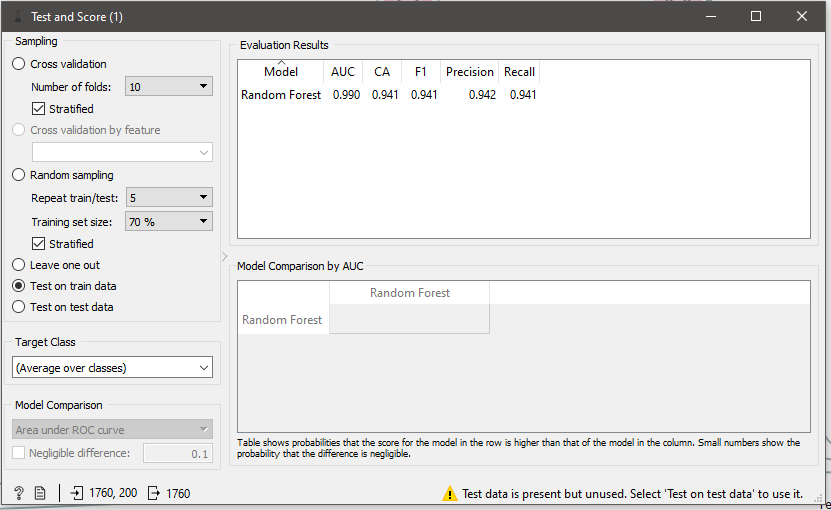
Hyperparameters: {Number of trees, Tree depth}

Results: {AUC, CA, F1, Precision, Recall} – The higher number the better

|  |  |
| --- | --- |
| **Hyperparameters** | **Results** |
| {1, 20} |  |
| {3, 20} |  |
| {6, 20} |  |
| {9, 20} |  |
| {12, 20} |  |
| {15, 20} |  |
| {17, 20} |  |
| {15, 1} |  |
| {15, 5} |  |
| {15, 10} |  |
| {15, 15} |  |
| {15, 20} |  |
| {15, 25} |  |
| {15, 30} |  |
| {15, 35} |  |

**Final solution**

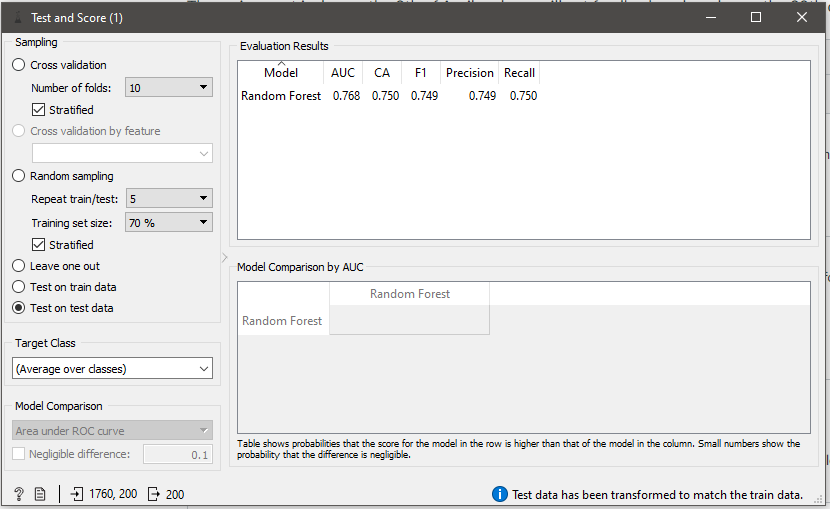
Judging from the cost function values shown in the grid search table above, I have decided that the best model to use for my data isthe Random Forest model with the hyper parameter of 15 trees with a depth of 25, as this gave the best output value. The following results are how well the model performed on all my training and validation data.

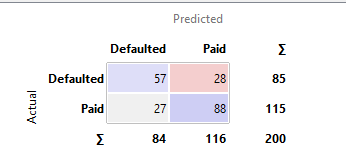


These scores show that with this model, we can get 0.990 out of 1 for the AOC value. AOC means area under the curve and is used to measure how well a classifier performs when classifying data points. The higher the AOC, the better the model is at classifying. CA is the classification accuracy of the model; it is the ratio of number of correct predictions over the total number of input samples. The F1 score also works like the CA value, it is used to measure how accurate the models’ tests are, the higher the score the more accurate predictions the model has made.

# Results and Errors

The following is the test results on the test data. As we can see the test data results are not as good as the result on the training data which can normally happen when testing this set, however this is not a problem if the difference is not too large. This difference is likely due to overfitting of the model causing it to not work as well on unseen data, meaning there is a problem with high variance. At this stage If we wanted to achieve a better score, we would have to go back some stages in the CRISP-DM methodology and think about whether we should change what variables are included, or if a different model could handle this data better and produce a low variance output.



The results of the predictions can be displayed as a confusion matrix.

The confusion matrix is a table that shows the values predicted by the model against the actual output values. For example, with this table we can see the predicted amount that would pay back is 28, and the amount the model predicted would pay back is 27. We can also see that the amount actually defaulted was 88 and the model predicted 57.This table helps us visualise how well the model has performed, and we can see that it is quite good at predicting successfully paid back loans, but not as good at predicted how many will fail to repay the loan.

For my system, a false negative is better is better than a false positive. A false negative would mean, for example, that we have predicted a customer will not pay back a loan when they actually would have. A false positive would mean that we predicted the customer to pay back the loan but instead it was defaulted. A false negative is better, a false positive would mean that the bank would not get their money back. Although too much false negatives could result in a loss of customers which would mean less money for the bank and could also be costly. This is not the case however as my model makes false negatives more often meaning they are losing customers more than they are losing money.