Data Mining Assessment - Tim Durand

Table of Contents

Out[93]: **NOTE**: The code for this notebook is, by default, hidden. To display the code cells, click here.

Abstract

The goal of this report, is to explore the key attributes, and develop several models, related to customer default payments in Taiwan. Data exploration, indicated that the financial attributes in the data, better described the target variable than the demographic attributes. Engineered features from the credit limit, balance amounts and payment status, were highlighted as some of the most important attributes when predicting a defaulting customer. The Random Forest model performed the best, from both an accuracy and cost perspective. However, the difference in accuracy for all the tuned models was negligible, therefore it was prudent to chose a less complex model; Logistic Regression. Looking at the cost of the models, the Random Forest did substantially out perform the Decision Tree, this may be due to the averaging of many trees reducing the overall error, and was chosen as the final model. Both final models performed well on the test data. However, further To take this research further, further feature engineering steps could be explored along with some Neural Network models. Python was used throughout the task, and the libraries provided some intuitive features for model fitting, tuning and feature selection.

Introduction

The purpose of this report, is to explore a dataset related to customer banking records in the month of September 2005, from a large Taiwanese bank. Within the supplied data, approximately 22% of customers defaulted on their payments; making this an unbalanced dataset. Below is a description of the attributes contained within the data:

Target Variable:

- DEFAULT: Default payment status

Predictor Variables:

- LIMIT_BAL: The amount of credit given to the customer (NT$).
- SEX: Customer gender, (Male or Female).
- EDUCATION: Customer education level (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown).
- MARRIAGE: Customer marriage status (Married, Single, Divorced, Widowed).
- AGE: Customer age in years.
- PAY_0: Repayment status one month prior (-2, -1 and 0 = no outstanding payment balance, 1=payment delay of one month, 2=payment delay for two months, etc, 9=payment delay of nine months and above).
- PAY_2 to PAY_6: Repayment status 2 to 6 months prior (same scale as PAY_0).
- BILL_AMT1 to BILL_AMT6: Bill amount on customer statement, 1 to 6 months prior (NT$).
- PAY_AMT1 to PAY_AMT6: Amount of payment, 1 to 6 months prior (NT$).

The objective of the assignment was to predict a customers default status as accurately as possible, in two distinct ways:

- Find a model with the best overall accuracy score (Equal costs for mis-classification).
- Find the model with the least overall cost, given the cost of a False Negative is 5 times greater than a False Positive.

The data was provided in two randomly divided sets, one for training and tuning each model and the other for the final evaluation of the best model assessed. CWdata_train.arff was used for Training and CWdata_test.arff for the final validation.

Data Exploration

Preliminary exploration and cleansing

Data profile: Training data

	Column	Max Length Value	Max Length	Min Length Value	Min Length	Null #	Null %	Data Type	Unique Values	Row Count	Mean
0	LIMIT_BAL	1e+06	9	50000	7	0	0	float64	80	25000	16758
1	SEX	b'M'	4	b'M'	4	0	0	string	2	25000	N/A
2	EDUCATION	b'2'	4	b'2'	4	0	0	string	7	25000	N/A
3	MARRIAGE	b'Divorced'	11	b'Single'	9	0	0	string	4	25000	N/A
4	AGE	49	4	49	4	0	0	float64	54	25000	35.473
5	PAY_0	-1	4	2	3	0	0	float64	11	25000	-0.0150
6	PAY_2	-1	4	2	3	0	0	float64	11	25000	-0.1352
7	PAY_3	-1	4	2	3	0	0	float64	11	25000	-0.1665
8	PAY_4	-2	4	2	3	0	0	float64	11	25000	-0.2218
9	PAY_5	-1	4	2	3	0	0	float64	10	25000	-0.2660
10	PAY_6	-2	4	2	3	0	0	float64	10	25000	-0.2897
11	BILL_AMT1	-165580	9	0	3	0	0	float64	19389	25000	51143.6
12	BILL_AMT2	223205	8	0	3	0	0	float64	19078	25000	49093.2
13	BILL_AMT3	1.66409e+06	9	0	3	0	0	float64	18796	25000	47020.6
14	BILL_AMT4	-170000	9	0	3	0	0	float64	18395	25000	43151.5
15	BILL_AMT5	205730	8	0	3	0	0	float64	17964	25000	40306.9
16	BILL_AMT6	-209051	9	0	3	0	0	float64	17611	25000	38795.1
17	PAY_AMT1	100467	8	0	3	0	0	float64	7077	25000	5636.15
18	PAY_AMT2	1.21547e+06	9	0	3	0	0	float64	7058	25000	5967.8
19	PAY_AMT3	130000	8	0	3	0	0	float64	6709	25000	5191.67
20	PAY_AMT4	138355	8	0	3	0	0	float64	6201	25000	4892.38
21	PAY_AMT5	100486	8	0	3	0	0	float64	6101	25000	4753.7
22	PAY_AMT6	232972	8	0	3	0	0	float64	6175	25000	5249.19
23	DEFAULT	b'Yes'	6	b'No'	5	0	0	string	2	25000	N/A

One can see from the initial profiling of the data, in the above table, that there are 24 columns and 25000 observations. None of the features have missing values, and there are two data types: string and float.

The string columns are encoded and therefore required decoding using a UTF-8 decoder.

Cleansed String Columns

	SEX	EDUCATION	MARRIAGE	DEFAULT
0	M	2	Married	Yes
1	F	2	Married	No
2	F	2	Married	No

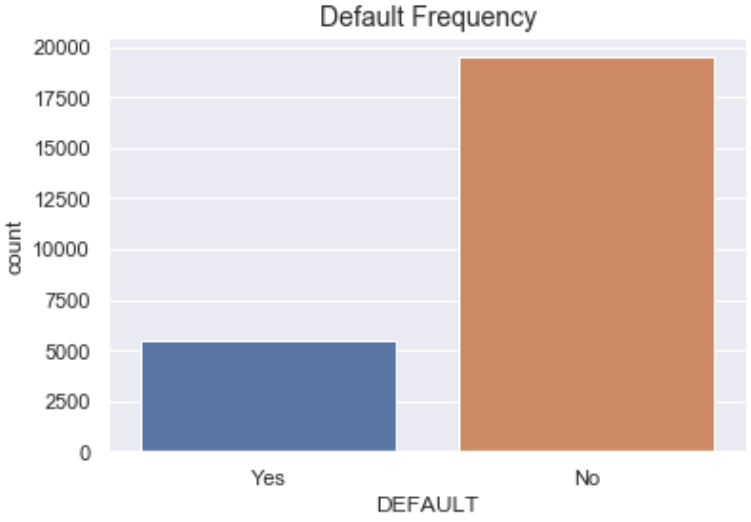
```
Initial unique values in EDUCATION:  ['2' '3' '1' '0' '5' '4' '6']
```

Based on the data description, the EDUCATION variable required some cleansing. Numerical values were converted to a description of the education level, to provide better interpretability. Attribute values variables 4,5 and 6 were grouped together to form Others, as a specific education level is unknown for all these classes. An attribute value of 0 was not present in the description, but was present in the data, this value was also set to Others.

```
Post cleanse unique values in EDUCATION:  ['University' 'High_School' 'Graduate
_School' 'Others']
```

Attribute values less than 0 in the PAY_n columns, are equivalent to 0; therefore all values less than 0 was set to 0.

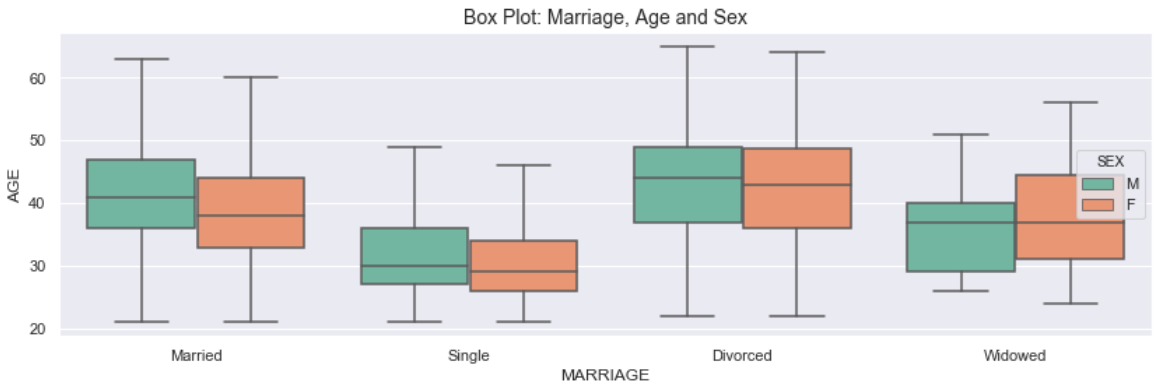
Class Imbalance

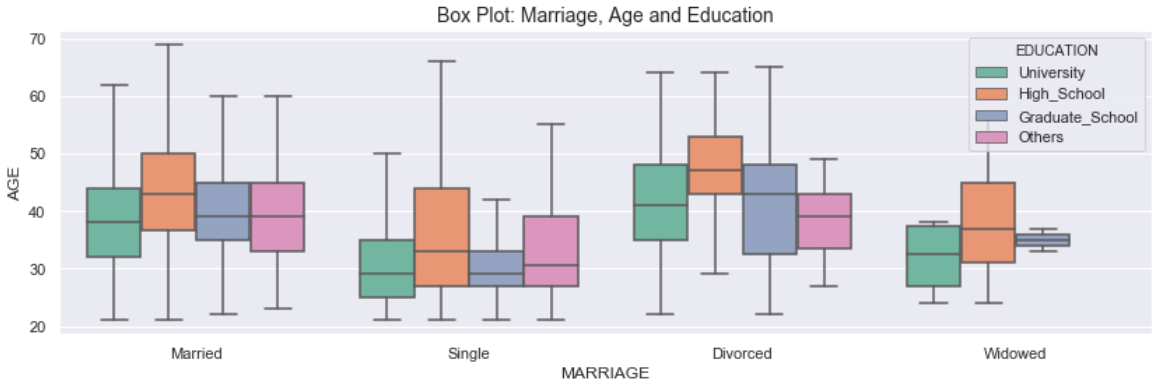


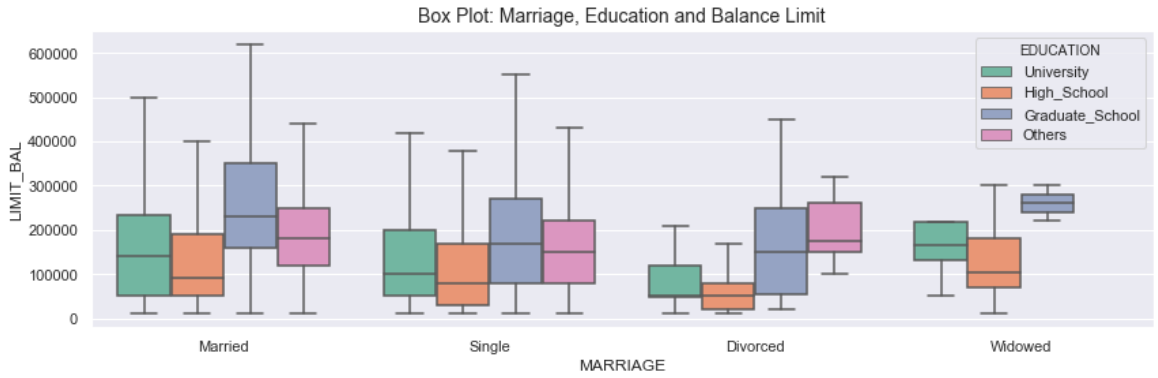

```
Percentage of Defaults:  22.06 %
```

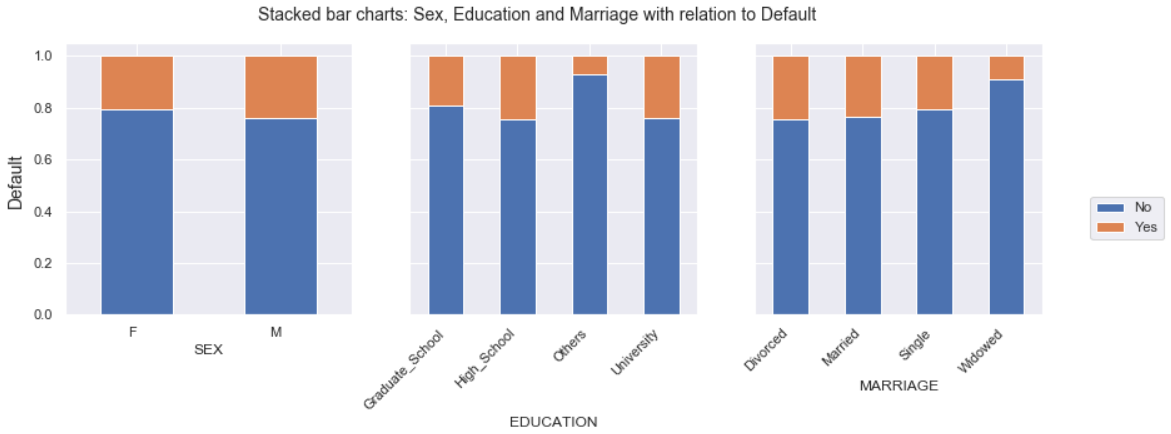
The above chart shows the frequency of the target attribute; the classes are imbalanced, with a ratio of 78:22.

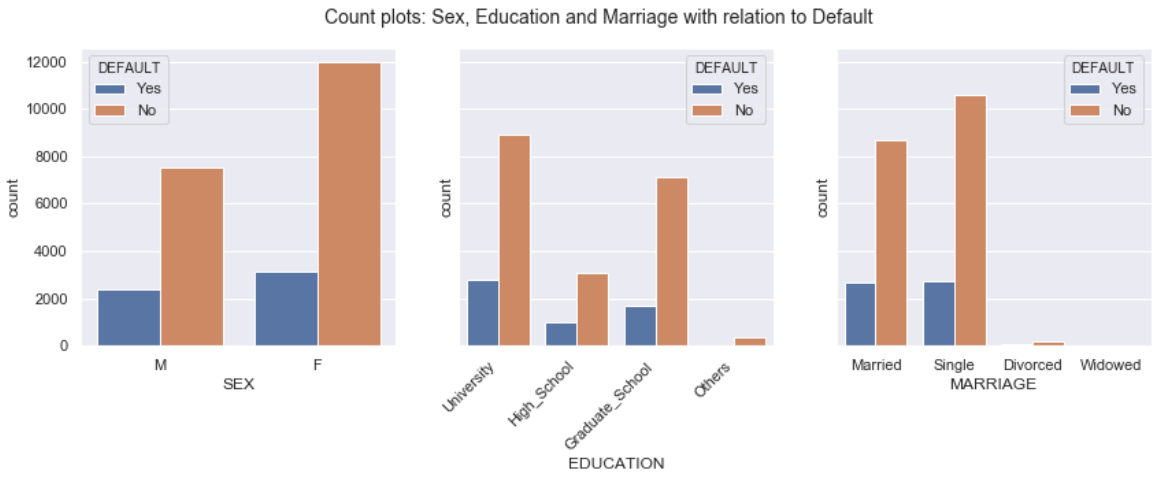
Demographic Analysis


Box Plot: Marriage, Age and Sex
AGE
60
50
40
30
20
SEX
M
F
Married
Single
Divorced
Widowed
MARRIAGE


Box Plot: Marriage, Age and Education
AGE
70
60
50
40
30
20
EDUCATION
University
High_School
Graduate_School
Others
Married
Single
Divorced
Widowed
MARRIAGE

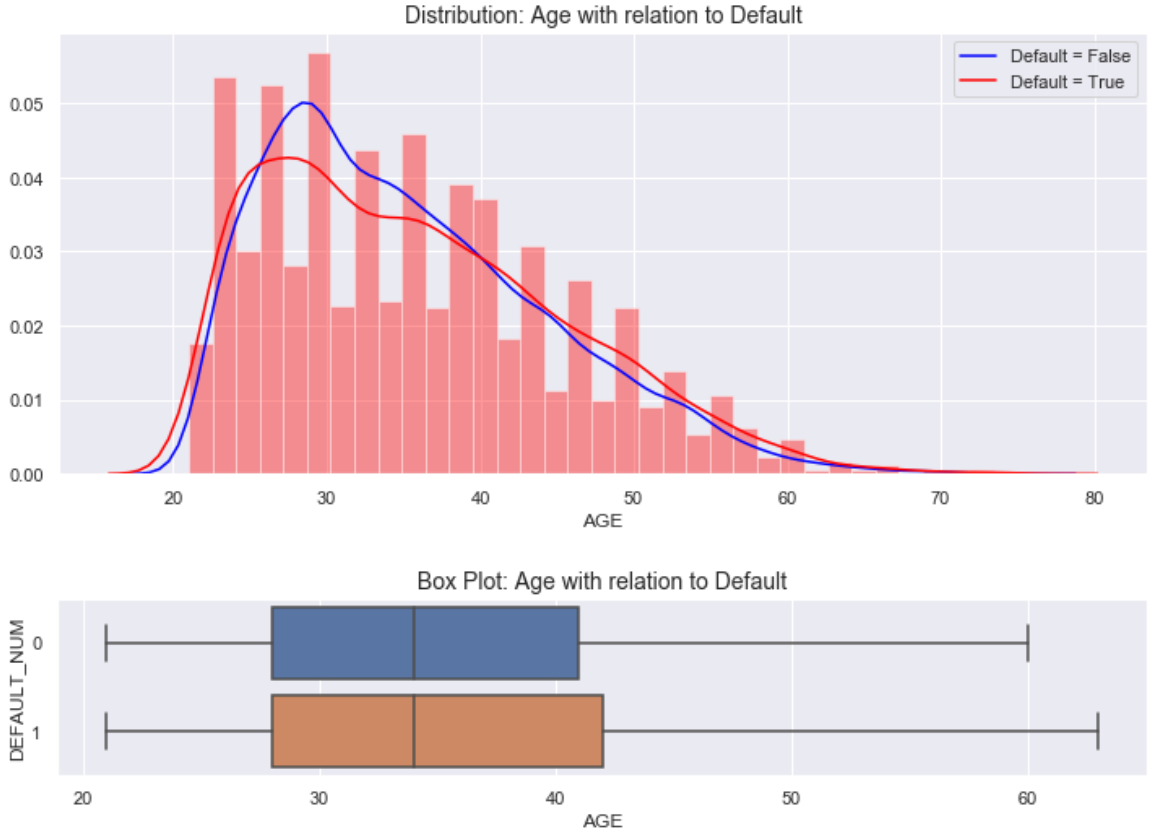

Box Plot: Marriage, Education and Balance Limit
LIMIT_BAL
600000
500000
400000
300000
200000
100000
0
EDUCATION
University
High_School
Graduate_School
Others
Married
Single
Divorced
Widowed
MARRIAGE


Stacked bar charts: Sex, Education and Marriage with relation to Default
Default
1.0
0.8
0.6
0.4
0.2
0.0
F
M
SEX
Graduate_School
High_School
Others
University
EDUCATION
Divorced
Married
Single
Widowed
MARRIAGE
No
Yes



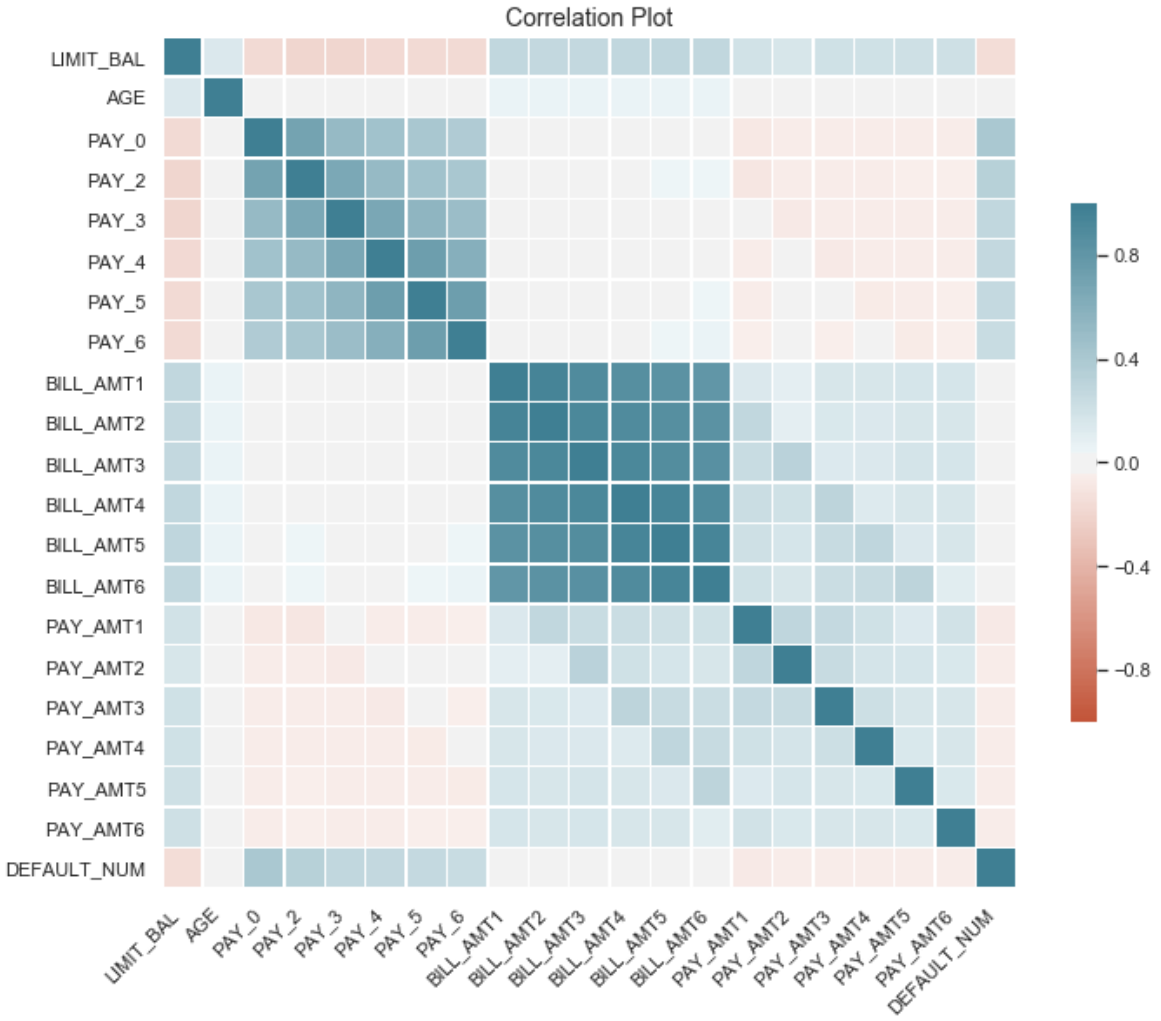
Sex and Marriage are nominal variables and Education is ordinal. One can deduce from the first chart that single people tend to be younger than those who are married, divorced or widowed, and the females in each category tend to be younger than males. Customers with a High school education tend to be older than those with a University or Graduate education, this may be a result of more prevalent higher education in recent years.

Sex does not seem to be a good predictor of whether a customer will default on their payment because there is very little variation between the two classes. Education also seems a poor predictor, because the highest variation is in an under-represented class (others). With Marriage the highest variation of Default is in the widowed class but this class, along with divorced, is under-represented.

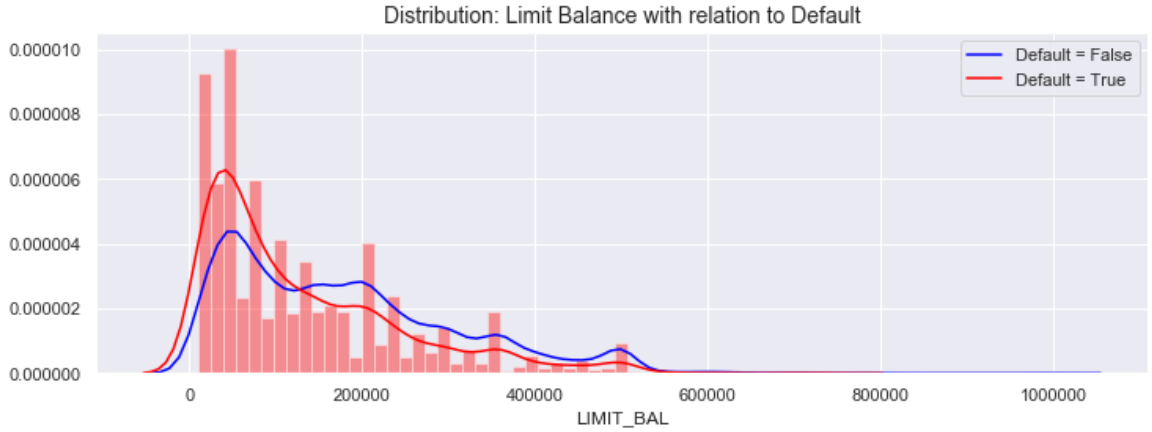


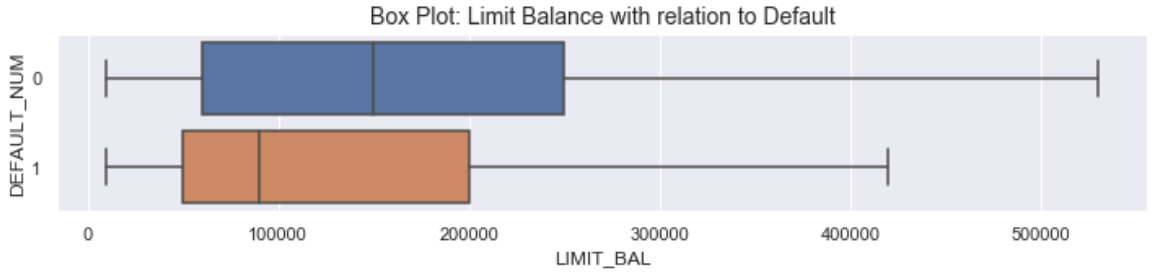
The mean age of customers who default is very similar to those who do not default and the distribution of the two groups is very similar so age does not look a good predictor

Financial Analysis

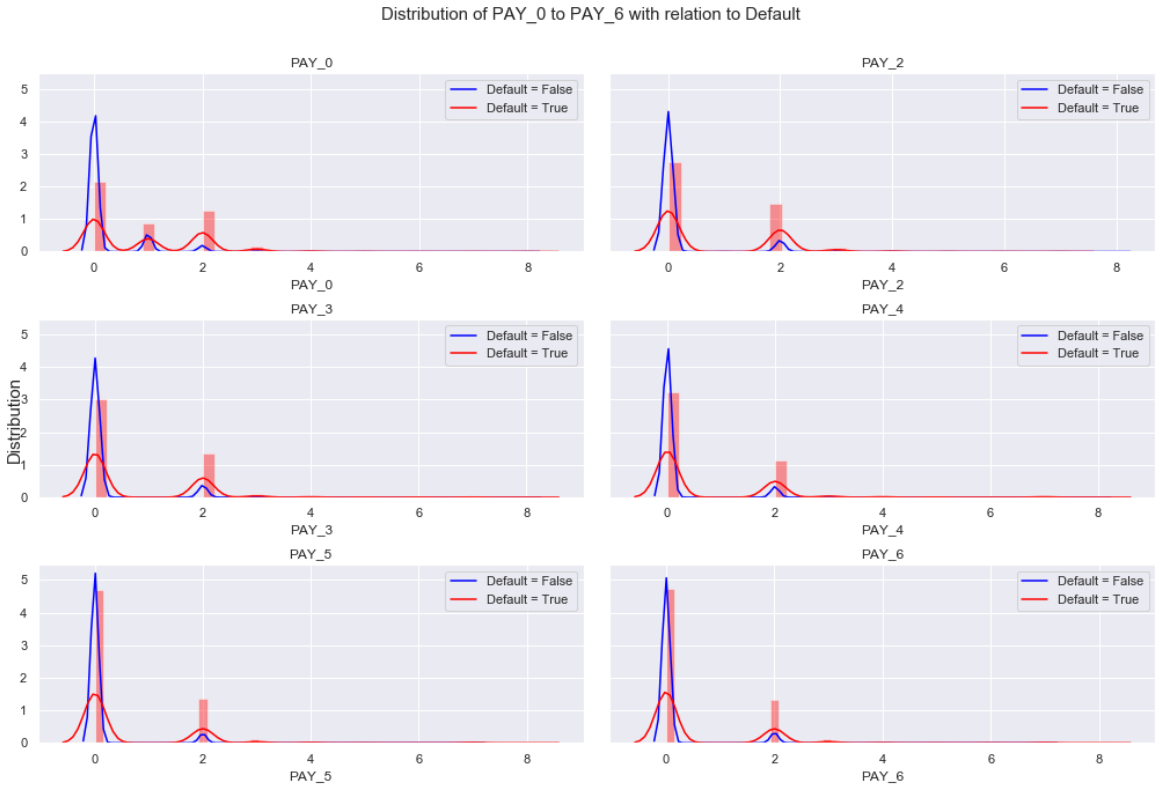


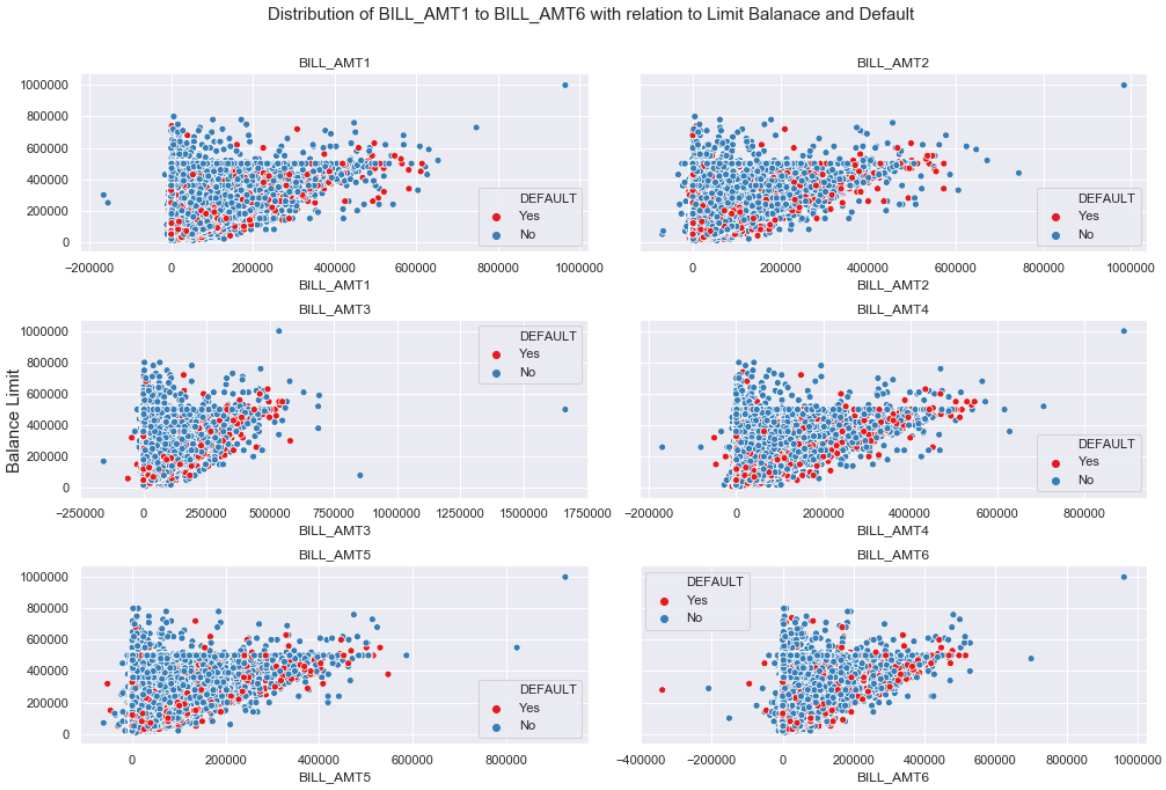
The above correlation plot is useful for identifying any relationships between parameters, including the target variable. Limit Balance is negatively correlated to customers who default, i.e as the balance increases the number of defaults decreases. This may mean that customers who have a good credit score, and have been good payers in the past, are given a larger limit because they can afford the re-payments. The PAY_n columns are correlated with each other, as well as BILL_AMT_n columns and the PAY_AMT_n columns. The correlation of the Pay and bill columns, infers that over time people are generally consistent with their payments and spending.
Based on the above the balance limit and Pay_0 seem to be the most descriptive parameters with relation to the target parameter.

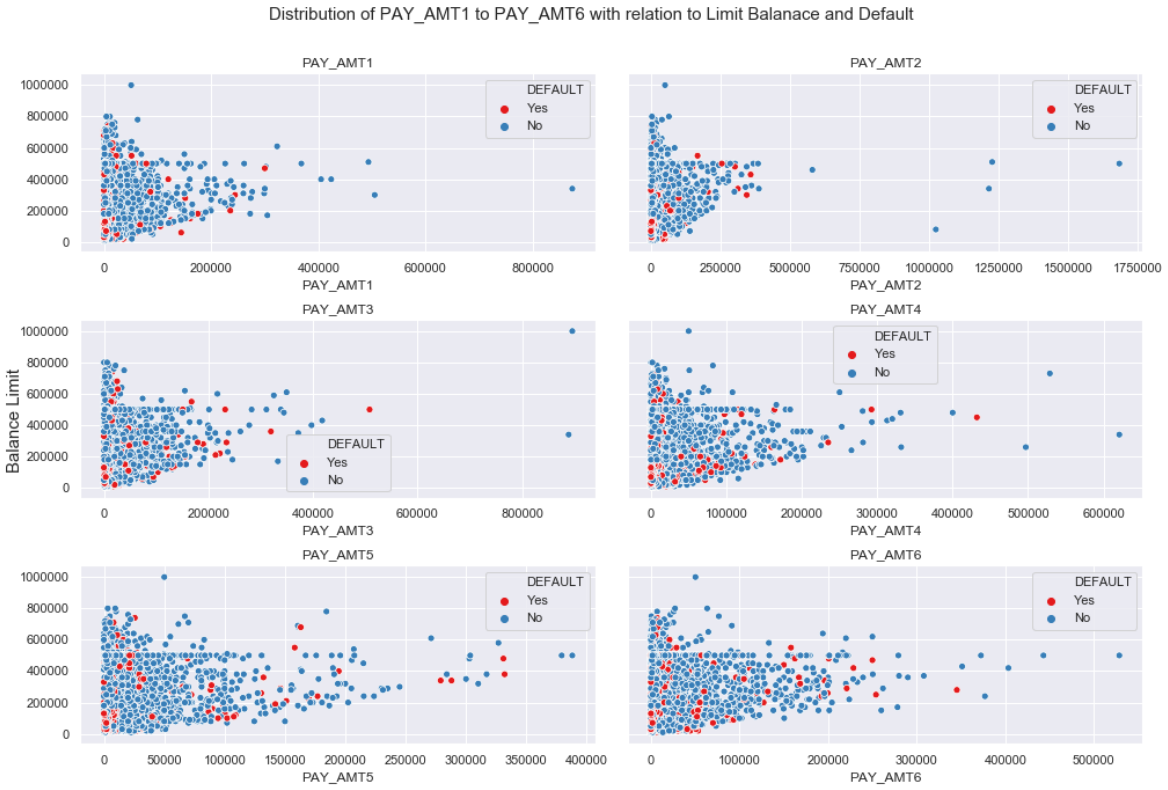




One can see from the charts above that the median balance limit is much higher for non-defaulting clients.

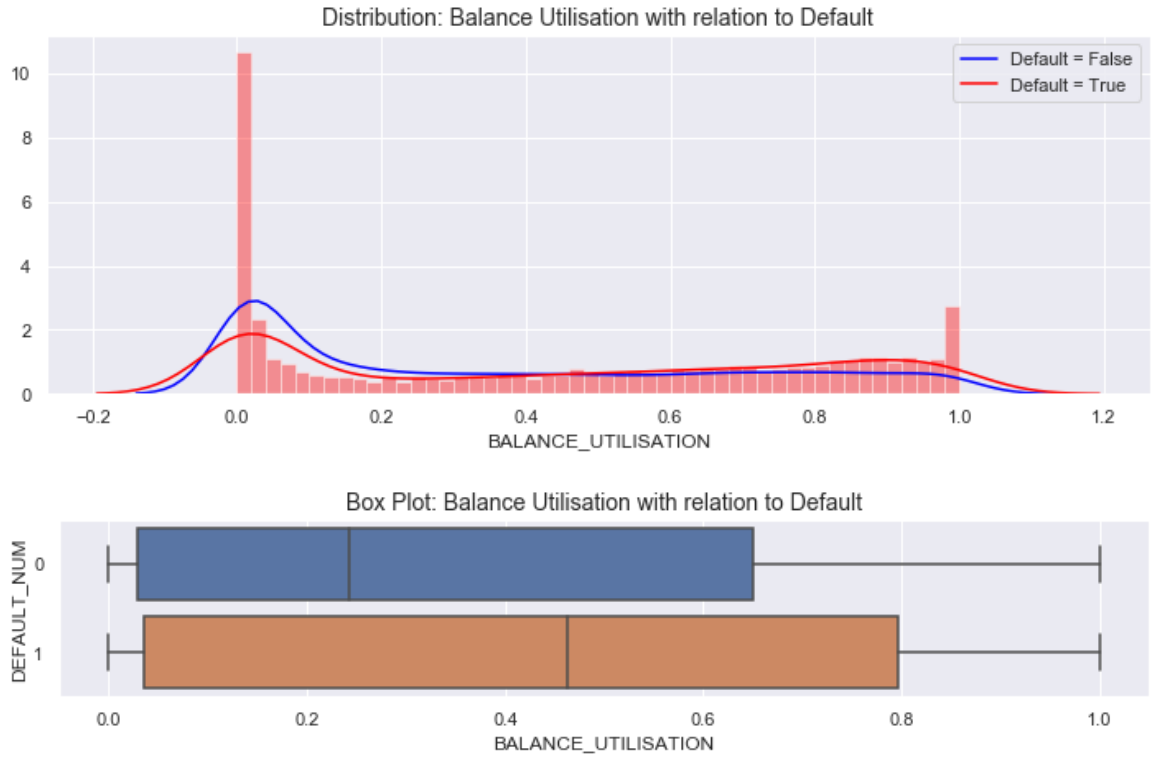

Distribution of PAY_0 to PAY_6 with relation to Default
PAY_0
PAY_2
PAY_3
PAY_4
PAY_5
PAY_6
Default = False
Default = True
Distribution


Distribution of BILL_AMT1 to BILL_AMT6 with relation to Limit Balanace and Default
BILL_AMT1
BILL_AMT2
BILL_AMT3
BILL_AMT4
BILL_AMT5
BILL_AMT6
DEFAULT
Yes
No
Balance Limit



One can see from the above plots of the Pay and bill columns, that within each group of parameters there is not much variation, meaning that perhaps these dimensions can be reduced. A mean value was created for each parameter group per customer.

The Bill plot suggests, that customers with a higher bill amount in relation to their limit, tend to be in the default class. The Balance utilisation parameter was created from the mean bill amount, as a percentage of their balance.



The newly created feature shows some predictive value do to the variance shown in the charts above between the two default classes.

Preprocessing for modelling

One Hot Encoding

Machine learning algorithms cannot work with categorical variables directly; the one hot encoding method was employed to convert categorical data to a binary attribute for each category.

Scaling

The data contains features with varying magnitudes and units; because many algorithms use the Euclidean distance between points , this can cause a higher importance to be placed on attribute values with a higher magnitude. The numeric attributes were all therefore scaled using a min/max scaler. Tree based models do not require attribute scaling, but the scaling should not have a negative effect and this kept the data consistent across the models.

Training and Validation split

The data was then split in two, with 70% for training and 30% for validation. The file CWdata_test.arff was not used for training or validation, and reserved for final model prediction only.

```
Training Shape: X = (17500, 34) y = (17500,)
Validation Shape: X = (7500, 34) y = (7500,)
```

Data profile: Independent Columns

	Column	Max Length Value	Max Length	Min Length Value	Min Length	Null #	Null %	Data Type	Unique Values
0	BALANCE_UTILISATION	0.000219231	22	0	3	0	0	float64	23069
1	SEX_F	0	1	0	1	0	0	uint8	2
2	SEX_M	1	1	1	1	0	0	uint8	2
3	EDUCATION_Graduate_School	0	1	0	1	0	0	uint8	2
4	EDUCATION_High_School	0	1	0	1	0	0	uint8	2
5	EDUCATION_Others	0	1	0	1	0	0	uint8	2
6	EDUCATION_University	1	1	1	1	0	0	uint8	2
7	MARRIAGE_Divorced	0	1	0	1	0	0	uint8	2
8	MARRIAGE_Married	1	1	1	1	0	0	uint8	2
9	MARRIAGE_Single	0	1	0	1	0	0	uint8	2
10	MARRIAGE_Widowed	0	1	0	1	0	0	uint8	2
11	MEAN_BILL_AMT_scaled	0.0615988	20	1	3	0	0	float64	22987
12	MEAN_PAY_scaled	0.0277778	20	0	3	0	0	float64	35
13	MEAN_PAY_AMT_scaled	1.06268e-06	22	0	3	0	0	float64	16589
14	PAY_0_scaled	0.125	5	0	3	0	0	float64	9
15	PAY_2_scaled	0.625	5	0	3	0	0	float64	9
16	PAY_3_scaled	0.375	5	0	3	0	0	float64	9
17	PAY_4_scaled	0.375	5	0	3	0	0	float64	9
18	PAY_5_scaled	0.375	5	0	3	0	0	float64	8
19	PAY_6_scaled	0.625	5	0	3	0	0	float64	8
20	LIMIT_BAL_scaled	0.00606061	21	0	3	0	0	float64	80
21	AGE_scaled	0.0185185	20	0.5	3	0	0	float64	54
22	BILL_AMT1_scaled	0.00938597	20	0	3	0	0	float64	19389
23	BILL_AMT2_scaled	0.0377495	20	0	3	0	0	float64	19078
24	BILL_AMT3_scaled	0.0610189	20	0	3	0	0	float64	18796
25	BILL_AMT4_scaled	0.178083	19	1	3	0	0	float64	18395
26	BILL_AMT5_scaled	0.0624131	20	0	3	0	0	float64	17964
27	BILL_AMT6_scaled	0.262499	19	1	3	0	0	float64	17611
28	PAY_AMT1_scaled	0.000373189	22	0	3	0	0	float64	7077
29	PAY_AMT2_scaled	0.00018762	22	0	3	0	0	float64	7058
30	PAY_AMT3_scaled	0.000366055	22	0	3	0	0	float64	6709
31	PAY_AMT4_scaled	0.0002657	22	0	3	0	0	float64	6201
32	PAY_AMT5_scaled	1.03074e-05	22	0	3	0	0	float64	6101
33	PAY_AMT6_scaled	0.000429383	22	0	3	0	0	float64	6175

One can see from the above table that all values fall between 0 and 1.

Classification Models

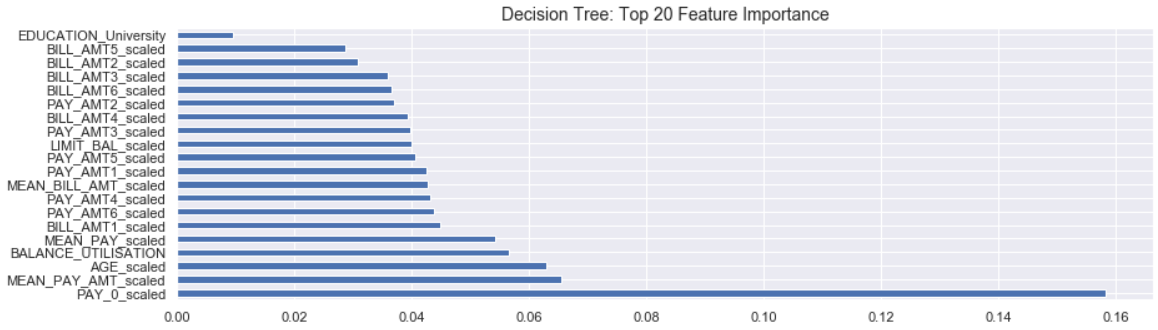
Four models were examined for each problem type (equal and unequal costs).

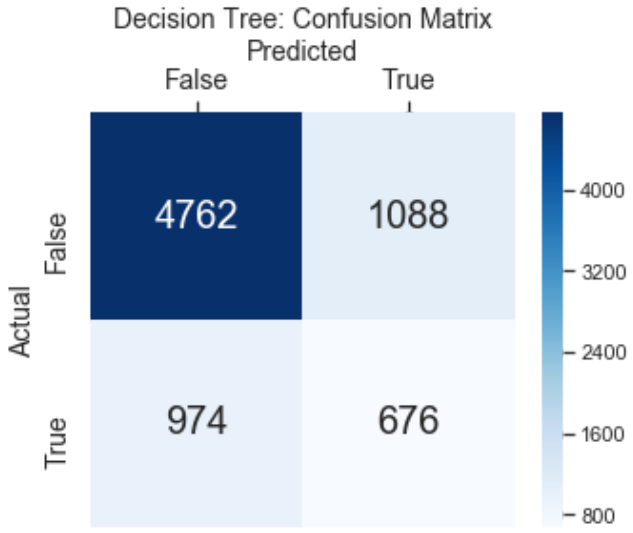
Equal Costs

The process for each model type, was to train and validate with default settings to get a baseline, before tuning the model, and comparing the performance with the original. The equal cost models were evaluated from an accuracy perspective. The sklearn library was used for all equal cost models.

Model 1: Decision Tree

The Decision Tree Classifier is a good choice as a first model, because it is relatively simple and can provide insight into the important features within the data. The Decision Tree Model from the sklearn library was used.



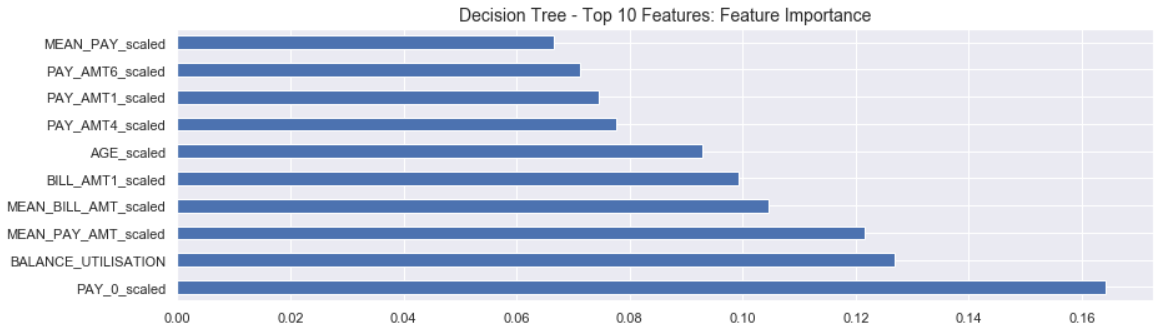


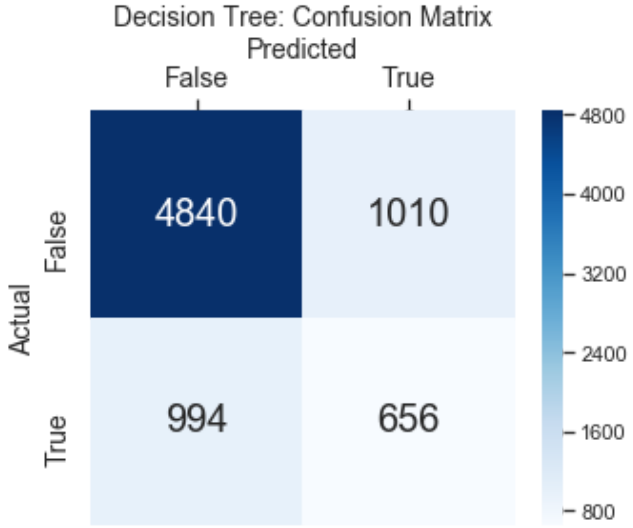

```
Classification Report
              precision    recall  f1-score   support

           0       0.83      0.81      0.82      5850
           1       0.38      0.41      0.40      1650

    accuracy                           0.73      7500
   macro avg       0.61      0.61      0.61      7500
weighted avg       0.73      0.73      0.73      7500
```

The above feature importance output, shows the Pay_0 is the most important feature, followed by the other financial columns; which was expected from the data exploration results. The relatively high importance of age was not expected. The categorical features have a very low feature importance. The accuracy of the model is 73%, which is the correctly classified items divided by all classified items, this seems like a good result but one has to remember that the data is not balanced. The confusion matrix gives a more detailed view of model performance.

Feature reduction

At this point only the 10 most important features were used to re-train and predict.





	precision	recall	f1-score	support
0	0.83	0.83	0.83	5850
1	0.39	0.40	0.40	1650
accuracy			0.73	7500
macro avg	0.61	0.61	0.61	7500
weighted avg	0.73	0.73	0.73	7500

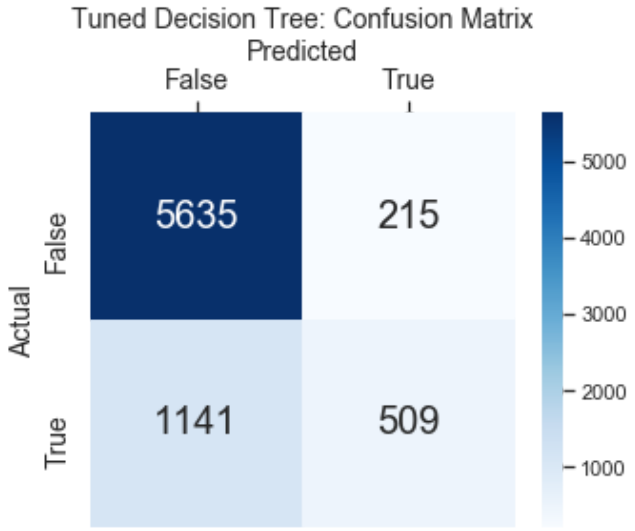
One can see from the above that the accuracy of the model does not decrease when only using 10 features. In fact, all models were trained and evaluated on all features, then just the top 10 above; the accuracy did not drop for any model.

Model 1: Tuning

A grid search was performed with some key hyperparameters to find the optimum settings. 3 fold cross validation was used as part of the function. Below is the list of hyperparameters and a description from the official documentation.

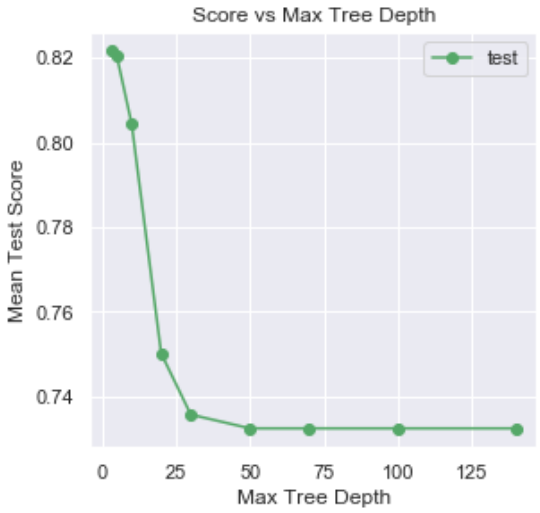
- max_depth (default=None): The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- max_features (default=auto): The number of features to consider when looking for the best split.
- min_samples_leaf (default=1): The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.
- min_samples_split (default=2): The minimum number of samples required to split an internal node.

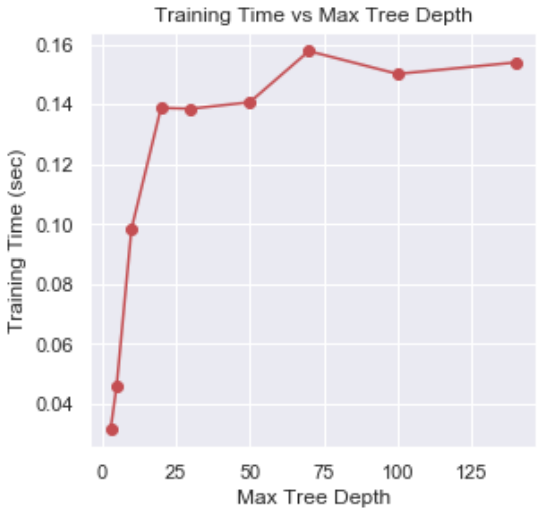
```
Best Parameters: {'max_depth': 3, 'max_features': None, 'min_samples_leaf': 1,
'min_samples_split': 2}
```



	precision	recall	f1-score	support
0	0.83	0.96	0.89	5850
1	0.70	0.31	0.43	1650
accuracy			0.82	7500
macro avg	0.77	0.64	0.66	7500
weighted avg	0.80	0.82	0.79	7500

The results above confirm that the tuned decision tree has increased the accuracy by 9%, a significant improvement on the default model. The default decision tree had similar False Positive and False Negative values, but the tuned model has a slightly higher number of False Negatives and a much lower number of False Positives.

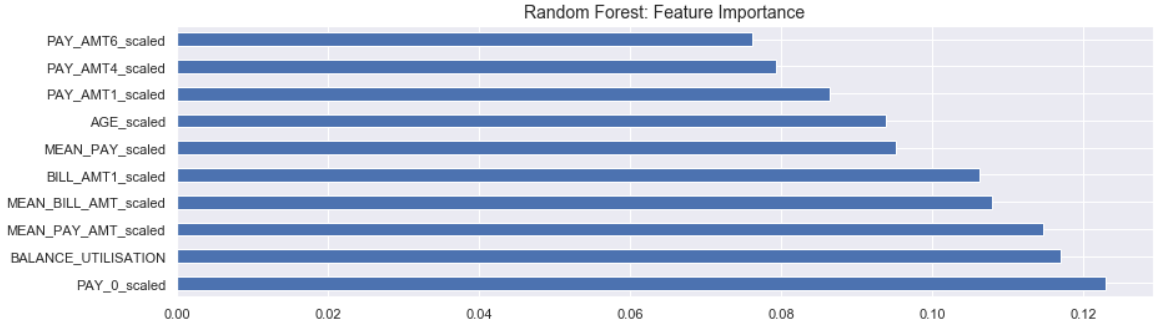


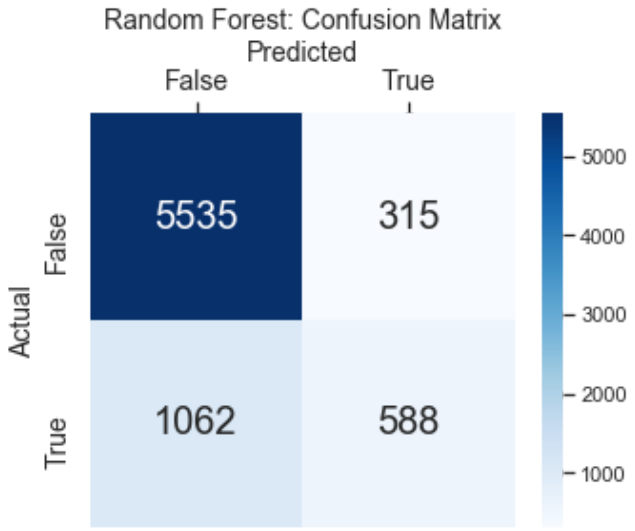


The maximum depth is probably the most important parameter of the tree, and can be used for pre-pruning. The grid search was repeated with the optimum values from the previous search and a variable maximum depth. One can see that the test score goes down, to a point with larger trees and the training time increases.

Model 2: Random Forest

The Random forest is the next logical step as it fits a number of decision trees on various sub-samples of the data, then averages the results to improve accuracy and reduce over-fitting. The Random Forest Classifier was used from the Sklearn library






```
Classification Report
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      5850
           1       0.65      0.36      0.46      1650

    accuracy                           0.82      7500
   macro avg       0.75      0.65      0.68      7500
weighted avg       0.80      0.82      0.80      7500
```

Pay_0 is still the most important feature, followed by some of the engineered columns: Balance Utilisation and the Mean Bill and Pay amounts. The accuracy of the default model is 82%, which is comparable to the tuned decision tree model. The confusion matrix is also more similar to the tuned decision tree than the default model, but the True Positive score has improved.

Model 2: Tuning

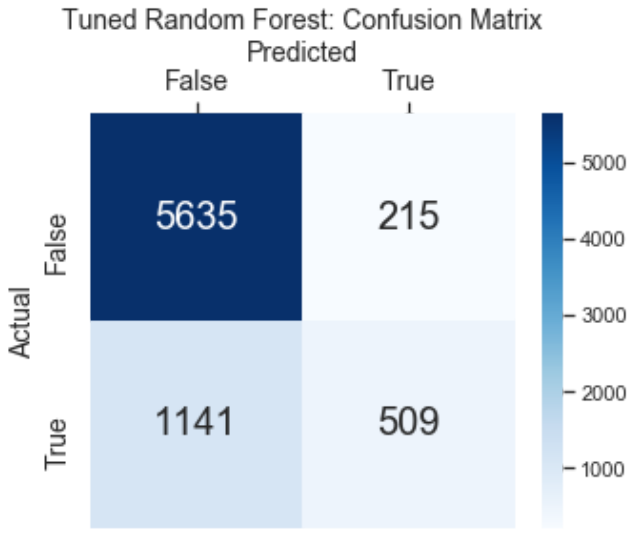
The grid search was composed of the same parameters as the decision tree with the n_estimators hyperparameter added:

- n_estimators (default=100) = The number of trees in the forest.

```
Best Parameters: {'max_depth': 10, 'max_features': 2, 'min_samples_leaf': 1, 'm
in_samples_split': 2, 'n_estimators': 100}
```

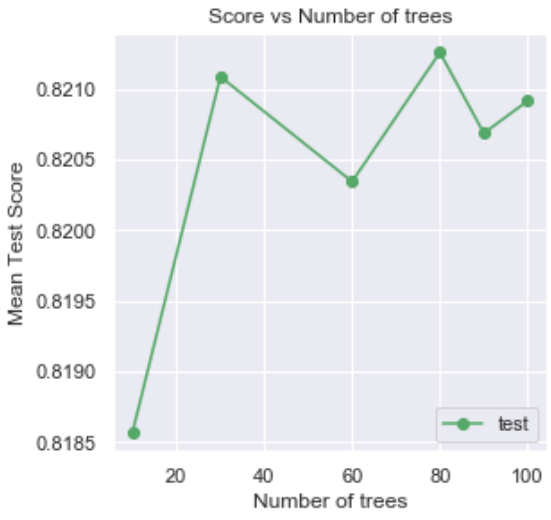
Tuned Random Forest: Confusion Matrix

Predicted	0	1
Actual		
0	5635	215
1	1141	509



	precision	recall	f1-score	support
0	0.84	0.96	0.89	5850
1	0.70	0.34	0.46	1650
accuracy			0.82	7500
macro avg	0.77	0.65	0.67	7500
weighted avg	0.81	0.82	0.80	7500

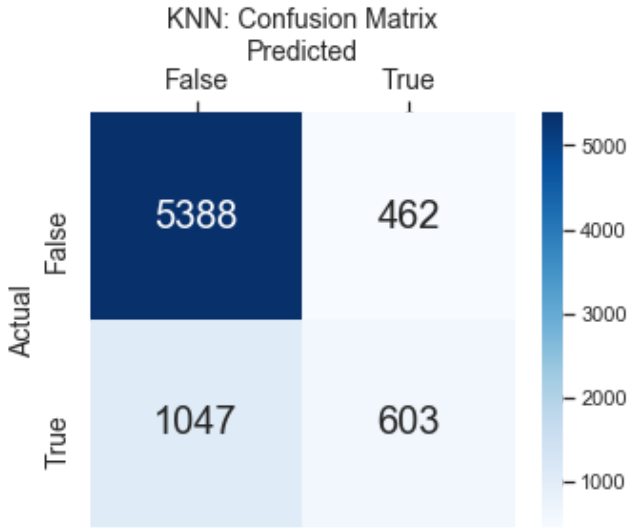
The grid search found that there was no substantial improvement with new parameters. The accuracy has not improved, and the confusion matrix has only slightly changed.



The above plot shows how the model score increases by around 1% when adding 20 more trees from 10 to 30 but there are some diminishing returns. The training time is a fairly linear increase with more trees.

Model 3: K Nearest Neighbors

The K Nearest Neighbors model is one of the more simple clustering models but often provides good results.



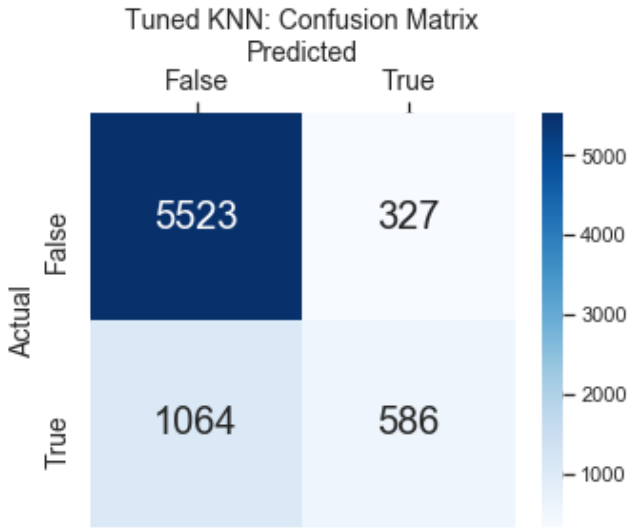

```
Classification Report
              precision    recall  f1-score   support

           0       0.84      0.92      0.88      5850
           1       0.57      0.37      0.44      1650

    accuracy                           0.80      7500
   macro avg       0.70      0.64      0.66      7500
weighted avg       0.78      0.80      0.78      7500
```

KNN performs better on a reduced number of features, this is because it is a lazy learner and requires more data with an increased number of features. The accuracy of the default model is 80%, around 2% better with the reduced feature set than the full set.

Model 3: Tuning

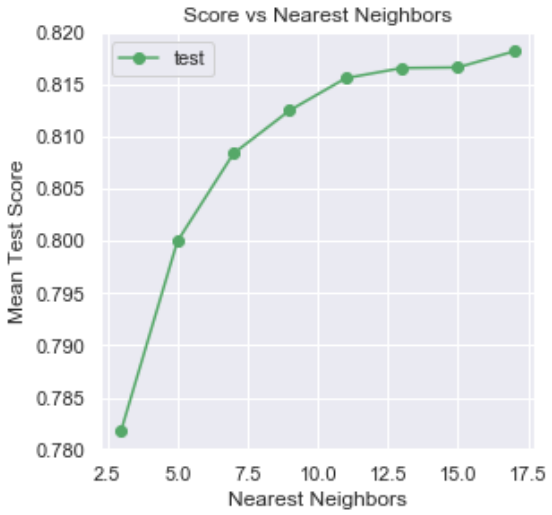
The grid search was composed of just the number of nearest neighbors(n_neighbors), the default value is 5.

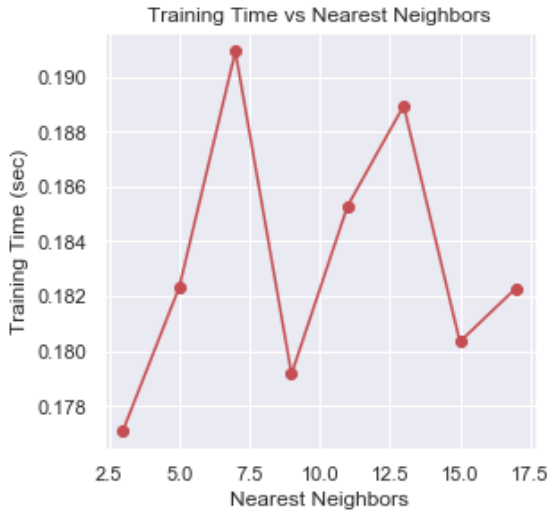
```
Best Parameters: {'n_neighbors': 17}
```



	precision	recall	f1-score	support
0	0.84	0.94	0.89	5850
1	0.64	0.36	0.46	1650
accuracy			0.81	7500
macro avg	0.74	0.65	0.67	7500
weighted avg	0.80	0.81	0.79	7500

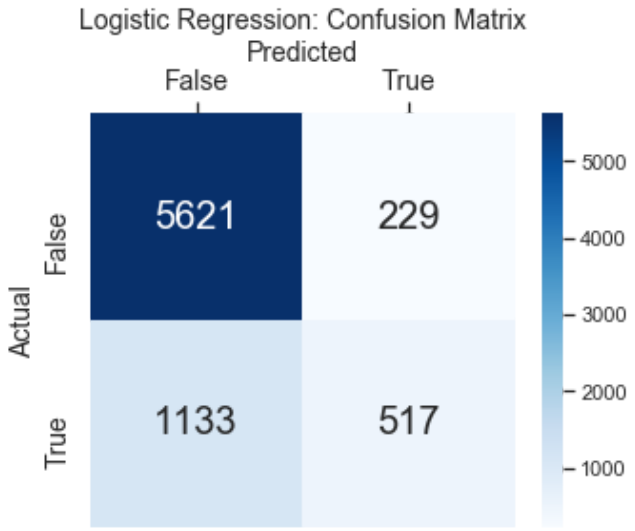
Increasing the number of neighbors increased the accuracy by around 1%. The recall is increased for non-defaulting customers, but reduced for defaulting customers; meaning the tuned model is slightly better for True Negatives, and worse for True Positives





Model 4: Logistic Regression

The logistic regression model, in simple terms, uses a logistic function to model a binary dependent variable, it is therefore a good candidate for this task.



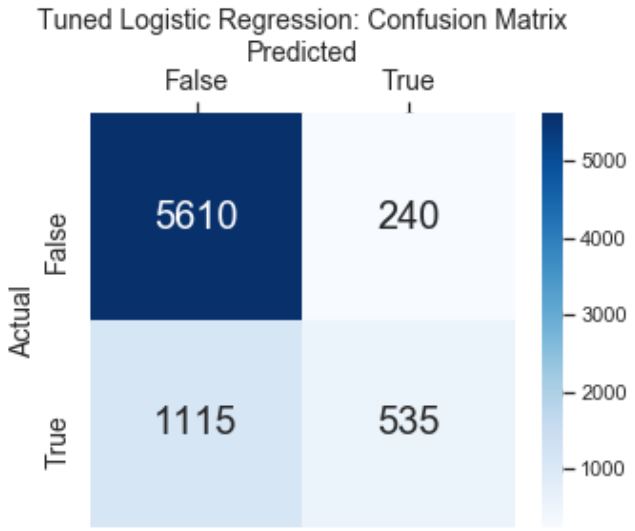

```
Classification Report
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      5850
           1       0.69      0.31      0.43      1650

    accuracy                           0.82      7500
   macro avg       0.76      0.64      0.66      7500
weighted avg       0.80      0.82      0.79      7500
```

Model 4: Tuning

Three parameters were used in the grid search, these are described below:

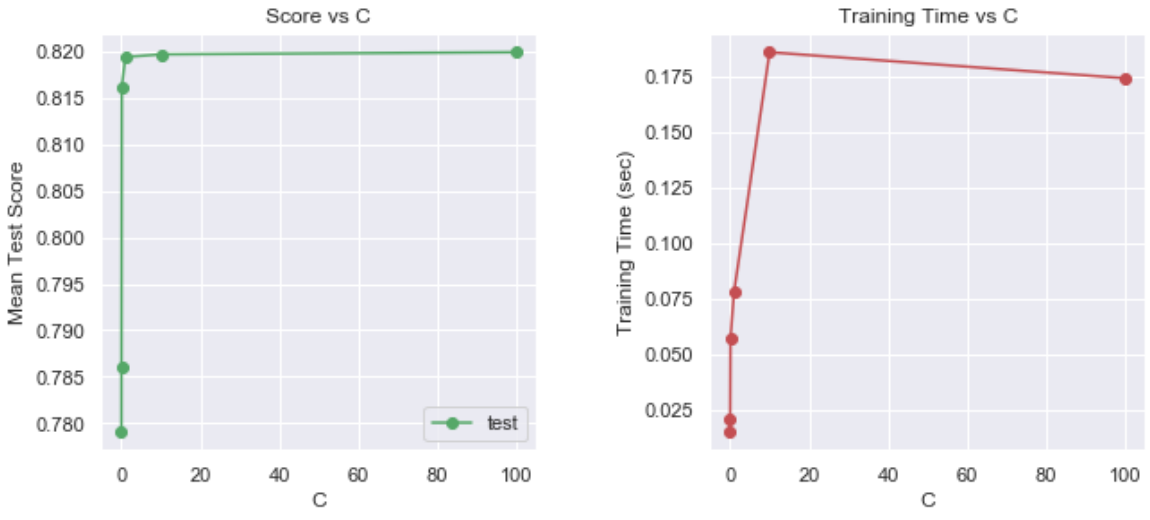
- penalty (default=l2): Used to specify the norm used in the penalization. The newton-cg, sag and lbfgs solvers support only l2 penalties. elasticnet is only supported by the saga solver. If none (not supported by the liblinear solver), no regularization is applied.
- C (default=1.0): Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.
- solver (default= lbfgs): Algorithm to use in the optimization problem. For small datasets, liblinear is a good choice, whereas sag and saga are faster for large ones.

```
Best Parameters: {'C': 100, 'penalty': 'l1', 'solver': 'liblinear'}
```

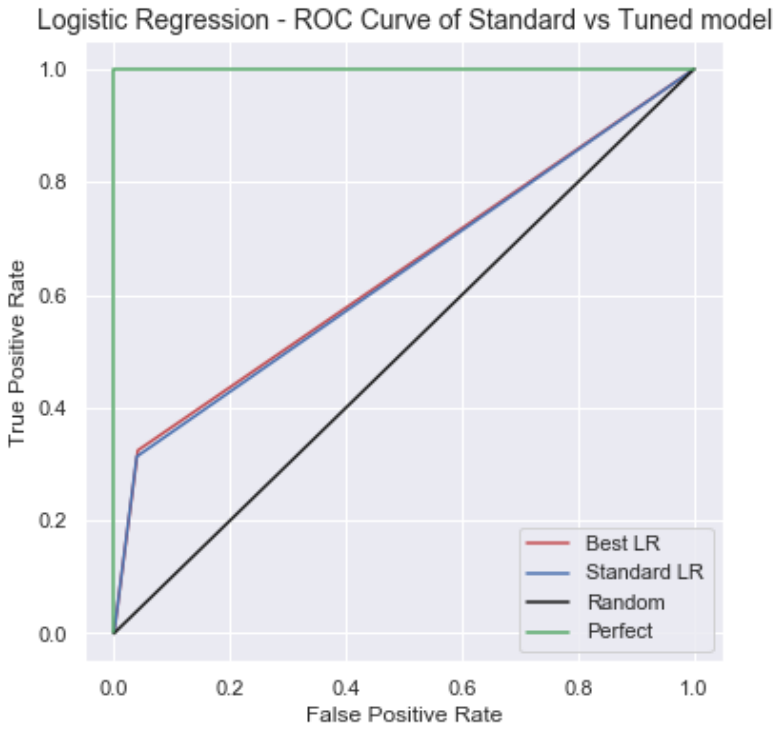


	precision	recall	f1-score	support
0	0.83	0.96	0.89	5850
1	0.69	0.32	0.44	1650
accuracy			0.82	7500
macro avg	0.76	0.64	0.67	7500
weighted avg	0.80	0.82	0.79	7500

This model provides results comparable to the Random forest, both in low False Positive numbers and overall accuracy.



Smaller values of C, increase the strength of regularization, this creates a simple model which can under-fit the data. Large values of C allow a model to increase it's complexity, and therefore, potential over-fitting can occur. One can see from the above variable values of C that the optimum level is relatively low, although the grid search identified 100 as the best value, increasing C beyond 10 does not increase the score vastly and may lead to over-fitting. Due to the simplicity of the model the training time is low compared to other models.



One can see from the above that only a small improvement in the ROC curve is obtained with the tuned model, ROC curves can be useful but they don't tell the whole story.

Unequal Costs

The unequal cost models were evaluated from an overall cost perspective rather than their respective accuracy scores. The imblearn library was used for cost sensitive modeling, the costcla library was also tried but the models did not seem to work well, and the documentation a little vague. The same model types were used in the balanced and unbalanced tasks. The process for each model type, was to compare the default cost sensitive model, with the default equal cost model before tuning the cost sensitive model and comparing the results with its equal costs, tuned, counterpart.

Cost Matrix

Cost Matrix

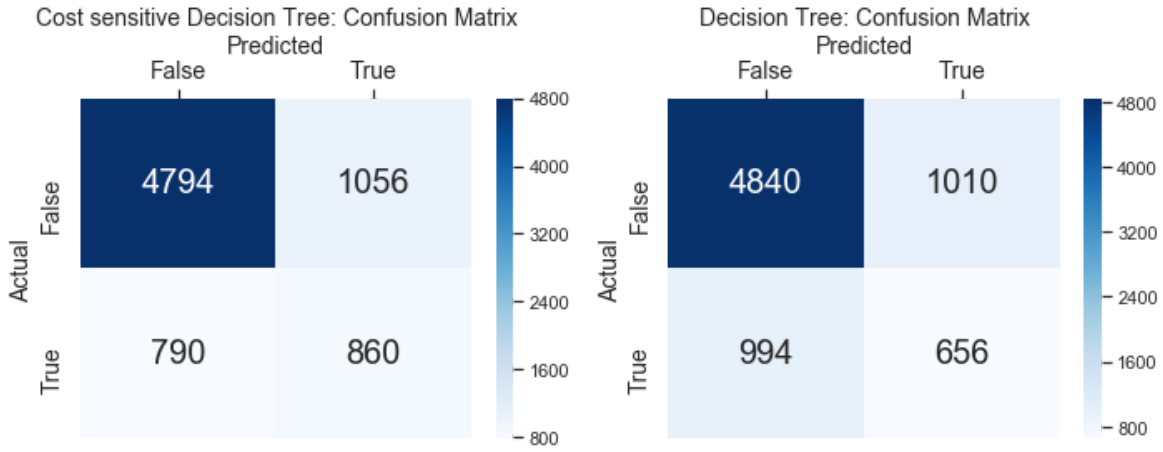
	False	True
False	0	1
True	5	0

The cost of misclassifying an actual defaulting customer, is 5 times higher than misclassifying a non-defaulting customer; this is expressed in the above Cost Matrix.

Model 1: Cost Sensitive Decision Tree

The imblearn library did not have a basic Decision Tree, KNN or Logistic regression model; however it has a bagging classifier, where any sklearn model can be used as the base model. Bagging refers to bootstrap aggregation, based on the principle that multiple weaker models can outperform a single stronger model. Variance is minimised which helps avoid over-fitting.

```
Cost Sensitive Decision Tree Cost =  5006
Decision Tree Cost =  5980
```

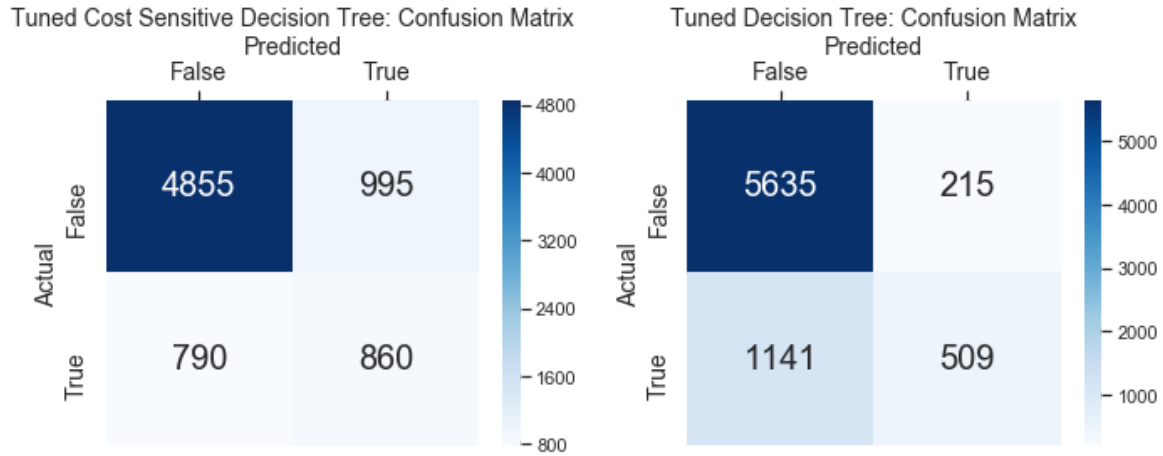


The cost sensitive decision tree reduced the number of False Negatives compared with the regular decision tree, making the cost of the model around 16% lower.

Model 1: Tuning

```
Best Parameters: {'base_estimator__max_depth': 30, 'base_estimator__max_feature
s': 4, 'base_estimator__min_samples_leaf': 1, 'base_estimator__min_samples_spli
t': 2}
```

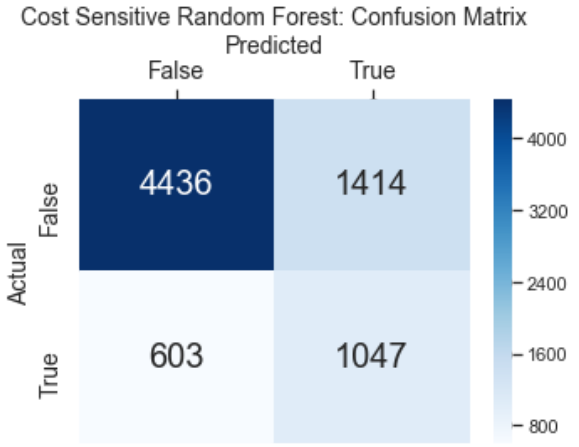
```
Tuned Cost Sensitive Decision Tree: Cost =  4945
Tuned Decision Tree: Cost =  5920
```

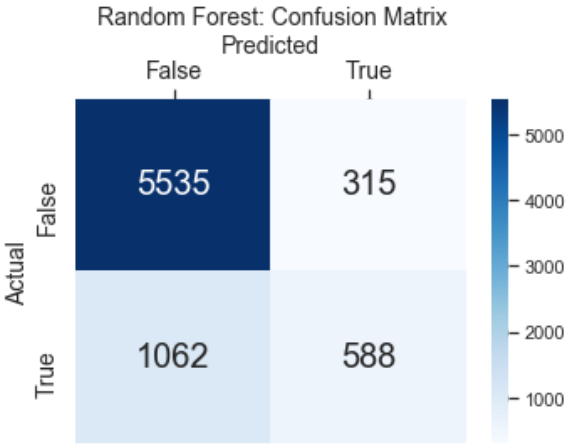


The same parameters, as the unbalanced model, were used for tuning the balanced model; but the optimum depth of the tree is 30 as opposed to 3. The increased tree depth could mean a more complex model is required, to better predict the under represented target class.

Model 2: Cost Sensitive Random Forest

```
Cost Sensitive Random Forest: Cost =  4429
Random Forest Cost =  5625
```

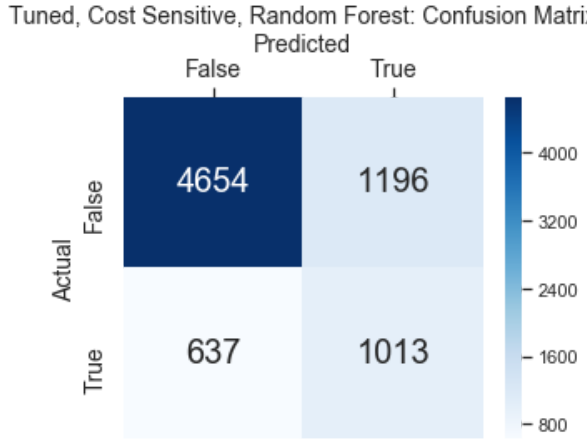


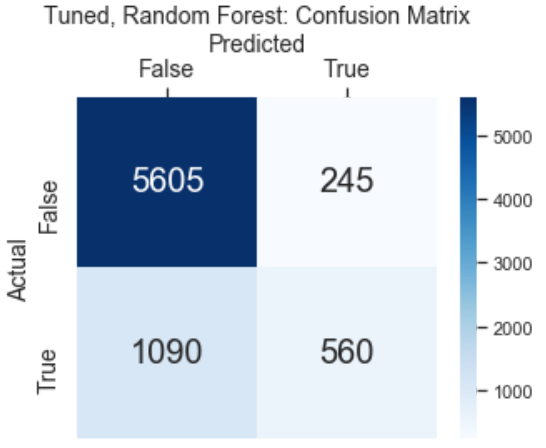


False Positives were much more prevalent in the cost sensitive random forest, but False Negatives, which carries a much higher cost penalty, were around 40% less leading to a lower cost model.

Model 2: Tuning

```
Best Parameters: {'max_depth': 10, 'max_features': 2, 'min_samples_leaf': 5, 'm
in_samples_split': 2, 'n_estimators': 100}

Tuned, Cost Sensitive, Random Forest: Cost =  4381
Tuned, Random Forest: Cost =  5625
```

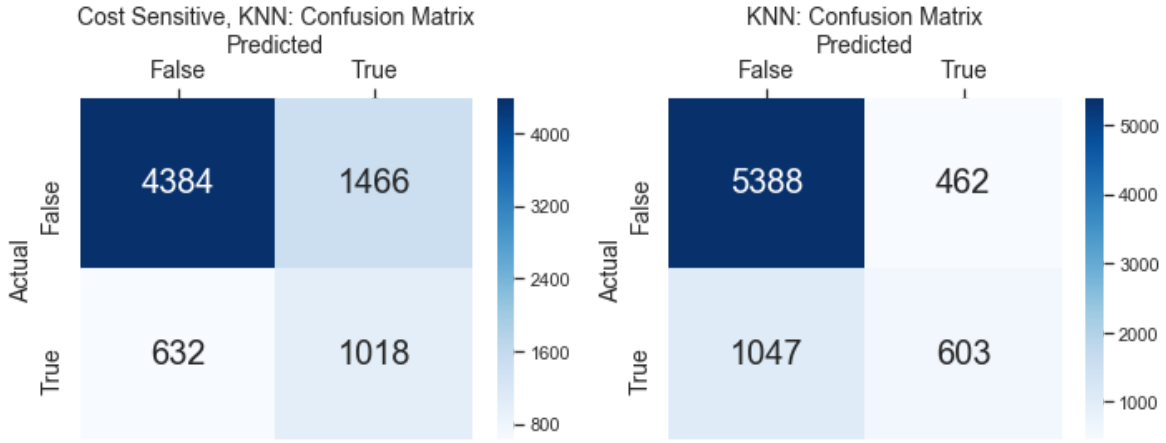




The best parameter set, output by the grid search, for the equal/unequal cost models, is very similar. The min_samples_leaf is higher; this parameter defines the minimum number of samples required to be at a leaf node; a higher value means an increase in bias and a decrease in variance, which may cause over-fitting.

Model 3: Balanced Bagging with KNN

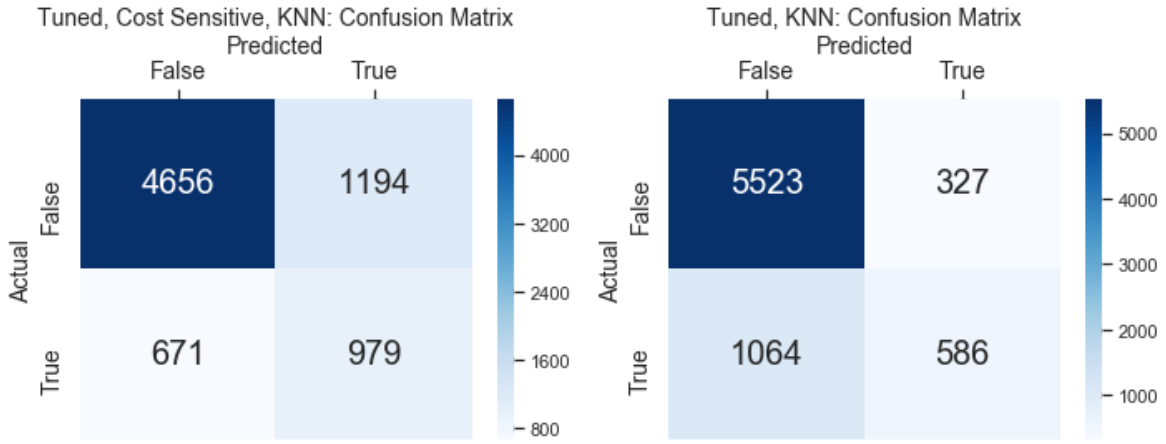
```
Cost Sensitive, KNN: Cost =  4626
KNN Cost =  5697
```



The confusion matrix above, shows that the accuracy of the balanced model is significantly lower than the original model, but the lower number of False Negatives, with a cost multiplier of 5, lowered the overall cost.

Model 3: Tuning

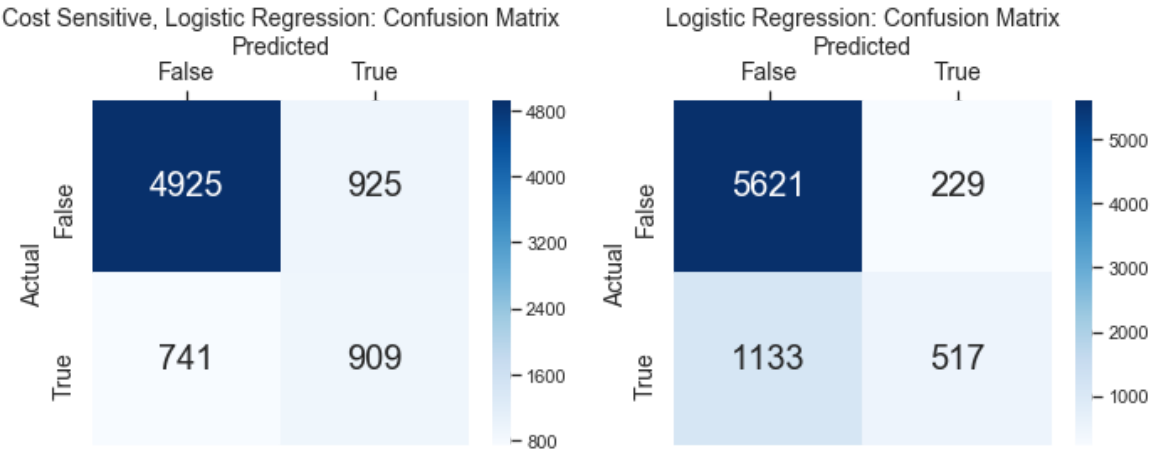
```
Best Parameters: {'base_estimator__n_neighbors': 17}

Tuned, Cost Sensative KNN: Cost =  4549
Tuned, KNN: Cost =  5647
```



The grid search, selected the same value of k as the equal costs model. The cost sensitive KNN model has reduced the number of False positives, meaning there is a good decrease in cost.

Model 4: Cost Sensitive Logistic Regression

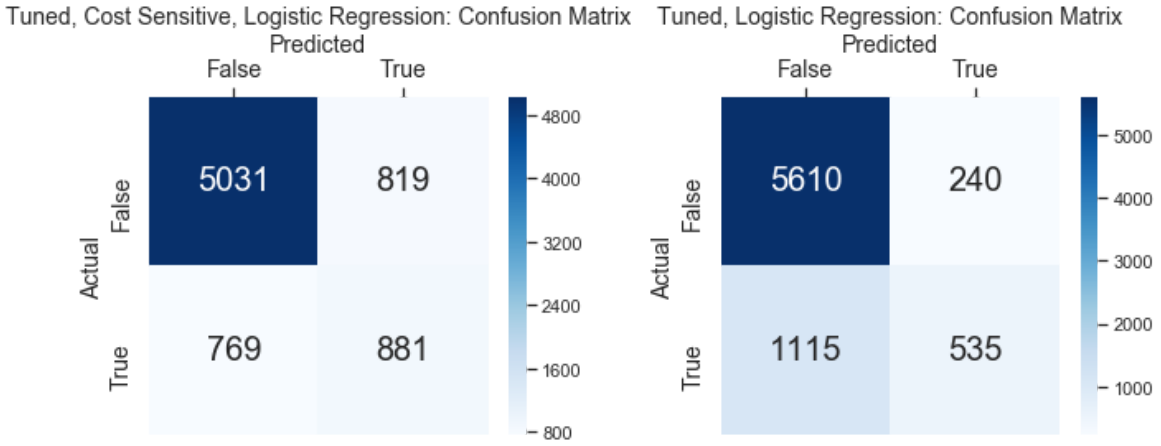
```
Cost Sensitive, Logistic Regression: Cost =  4630
Logistic Regression: Cost =  5894
```



Model 4: Tuning

```
Best Parameters: {'base_estimator__C': 0.1, 'base_estimator__penalty': 'l2', 'b
ase_estimator__solver': 'liblinear'}
```

```
Tuned, Cost Sensitive, Logistic Regression: Cost =  4664
Tuned, Logistic Regression: Cost =  5815
```



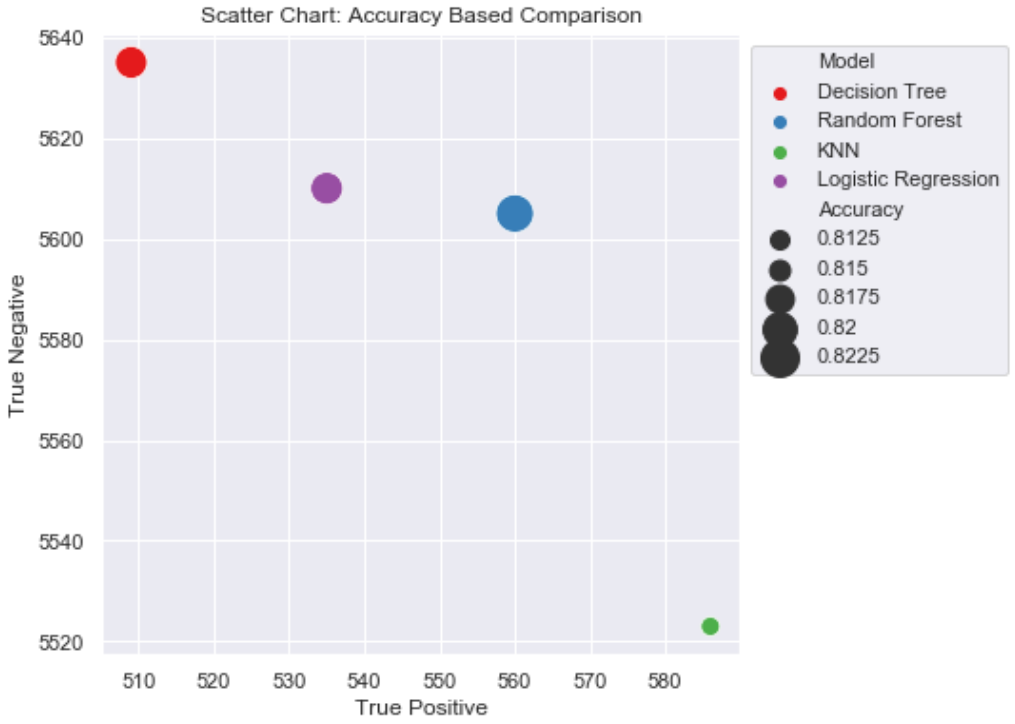
The default and tuned cost sensitive models, both reduced the number of False Negatives providing, a lower overall cost than the equal cost models. The cost for the default cost sensitive model, is actually lower than the tuned model, this is due to the metric used in the grid search for finding the best model.

Model Comparison

Equal Costs

Table: Accuracy Based Comparison

	Model	Accuracy	True Positive	True Negative
1	Random Forest	0.822	560	5605
3	Logistic Regression	0.819333	535	5610
0	Decision Tree	0.8192	509	5635
2	KNN	0.814533	586	5523



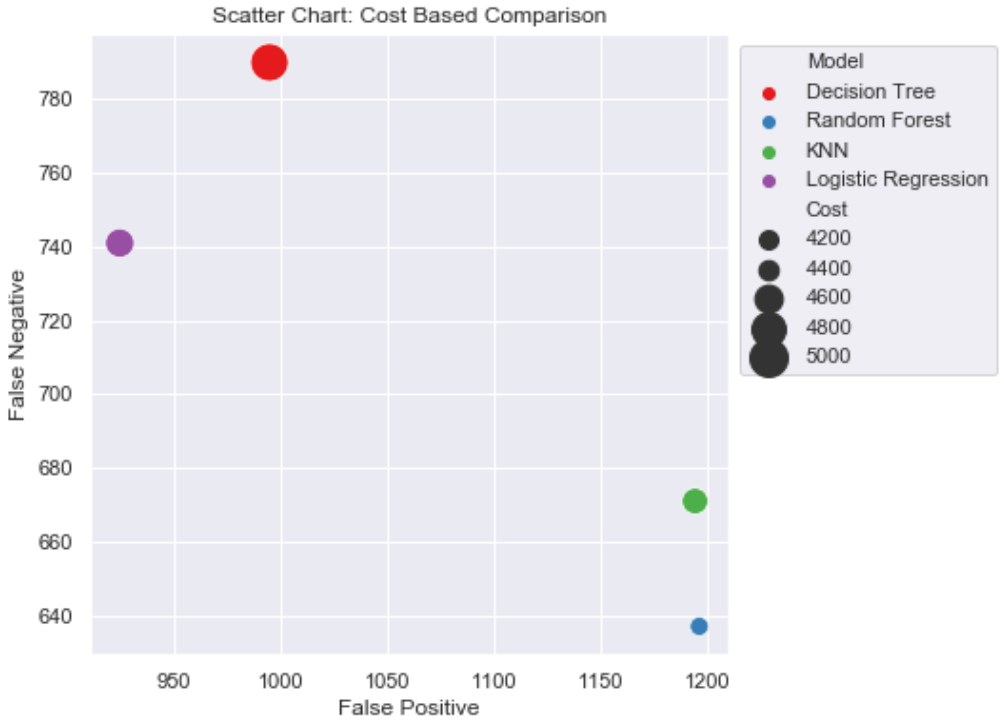
The accuracy, true positives, and true negatives for all tuned models are were collected and displayed above. The size of the point in the scatter chart denotes the accuracy, which is plot on the true positive/negative axes.

The information above, shows the best overall model from an accuracy perspective, is the Random Forest. The difference between the tuned models is relatively small, so it may be prudent to chose a simpler model from a computational perspective; Logistic Regression was chosen as the final model as it offered similar accuracy but less complexity. A simpler model may lead to a better result on the unseen test data due to less risk of over-fitting.

Unequal Costs

Table: Cost Sensitive Comparison

	Model	Cost	False Positive	False Negative	Accuracy
1	Random Forest	4381	1196	637	0.7556
2	KNN	4549	1194	671	0.751333
3	Logistic Regression	4630	925	741	0.777867
0	Decision Tree	4945	995	790	0.762



The cost, accuracy, false positives, and false negatives for all tuned, cost sensitive models are displayed above. The size of the point in the scatter chart denotes the cost, which is plot on the false positive/negative axes.

The best model from a cost perspective, is again, the Random Forest. The un-tuned Logistic Regression model is shown above, as this had less overall cost than the tuned model, it also had the highest accuracy of all models. KNN had a similar cost to the Random Forest but the training time was similar so the Random Forest chosen as a final model. If the experiment was to be extended, a custom scoring function could be created, and used for grid searching hyperparameters. This may lead to better performing tuned models from a cost perspective.

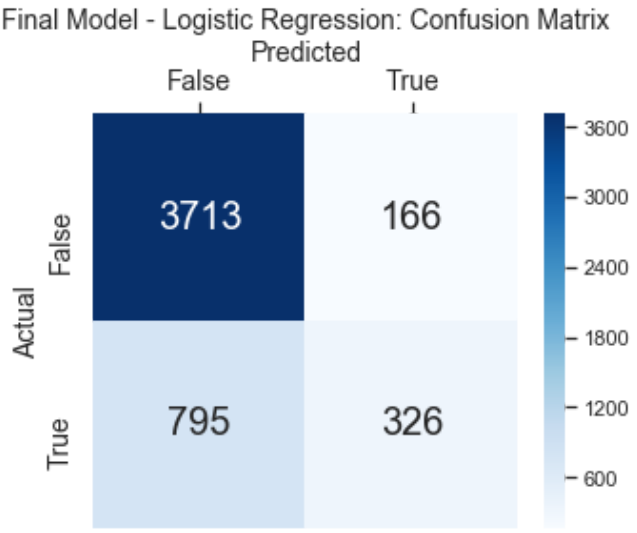
Best Model Prediction on Test Data

Data profile: Final Test Data - Independent Columns

	Column	Max Length Value	Max Length	Min Length Value	Min Length	Null #	Null %	Data Type	Unique Values	Row Count
0	PAY_0_scaled	0.125	5	0	3	0	0	float64	9	5000
1	MEAN_PAY_AMT_scaled	0.00025663	22	0	3	0	0	float64	4004	5000
2	AGE_scaled	0.0517241	20	0	3	0	0	float64	53	5000
3	BALANCE_UTILISATION	0.000155556	22	0	3	0	0	float64	4726	5000
4	MEAN_PAY_scaled	0.030303	20	0	3	0	0	float64	30	5000
5	BILL_AMT1_scaled	0.0474008	20	1	3	0	0	float64	4329	5000
6	PAY_AMT6_scaled	1.13821e-05	22	0	3	0	0	float64	1878	5000
7	PAY_AMT4_scaled	0.000479229	22	0	3	0	0	float64	1915	5000
8	MEAN_BILL_AMT_scaled	0.0259842	20	0	3	0	0	float64	4757	5000
9	PAY_AMT1_scaled	4e-05	22	0	3	0	0	float64	2170	5000

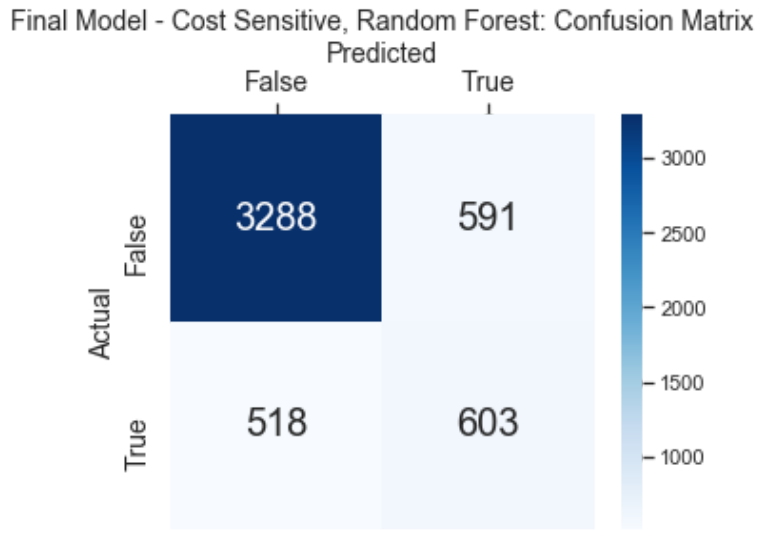
The CWdata_test.arff was imported and the pre-processing steps applied. A profile of the final test data is shown above, and one can see it is in the same format as the training data. The mean values, look similar to the training data, but there are only 5000 observations.

Equal Costs




```
Final Model - Logistic Regression: Accuracy =  0.8078

Classification Report
              precision    recall  f1-score   support

           0       0.82      0.96      0.89      3879
           1       0.66      0.29      0.40      1121

    accuracy                           0.81      5000
   macro avg       0.74      0.62      0.64      5000
weighted avg       0.79      0.81      0.78      5000
```

Unequal Costs




```
Final Model - Cost Sensitive Random Forest: Cost =  3181

Classification Report
              precision    recall  f1-score   support

           0       0.86      0.85      0.86      3879
           1       0.51      0.54      0.52      1121

    accuracy                           0.78      5000
   macro avg       0.68      0.69      0.69      5000
weighted avg       0.78      0.78      0.78      5000
```

Conclusion

Analysing at the equal costs results for the final model; the model generalised well to the previously unseen data, meaning the model did not over-fit the data. The accuracy is slightly less and the recall lower on the positive class than the training data. The accuracy results were fairly consistent across the different models explored, once optimum hyperparameters were chosen; perhaps different pre-processing steps could have increased performance.

The final cost sensitive model actually had a considerably lower cost than when fit to the training set, however it still predicted the negative class much better than the positive class. The unequal cost problem is more difficult to solve, than when considering equal costs, and there is less out-of-the-box functionality for cost sensitive learning in Python. A custom cost score that could be used in a grid search may be a way to improve results, also the use of neural networks could be explored.

If this experiment were to be completed again, the pipeline functionality in the sklearn library could be used, to make the process less onerous. However, the process has been both enlightening and enjoyable.

References

- Data Visualization using Seaborn: https://towardsdatascience.com/data-visualization-using-seaborn-fc24db95a850 (https://towardsdatascience.com/data-visualization-using-seaborn-fc24db95a850)
- Dummy Coding: The how and why: https://www.statisticssolutions.com/dummy-coding-the-how-and-why/ (https://www.statisticssolutions.com/dummy-coding-the-how-and-why/)
- Seaborn: API reference: https://seaborn.pydata.org/api.html (https://seaborn.pydata.org/api.html)
- Sklearn API reference: https://scikit-learn.org/stable/modules/classes.html (https://scikit-learn.org/stable/modules/classes.html)
- Decision Tree classification: https://www.datacamp.com/community/tutorials/decision-tree-classification-python (https://www.datacamp.com/community/tutorials/decision-tree-classification-python)
- Confusion matrix: https://medium.com/hugo-ferreiras-blog/confusion-matrix-and-other-metrics-in-machine-learning-894688cb1c0a (https://medium.com/hugo-ferreiras-blog/confusion-matrix-and-other-metrics-in-machine-learning-894688cb1c0a)
- Why Random Forest is My Favorite Machine Learning Model: https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706 (https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706)