Install relevant libraries

I use the pip python package manager to install the relevant modeules for this analysis. Modules already present in the Anaconda distribution are optional to install.

Imports

In this code block, I import the modules that will be used for the analysis. These include modules for data analysis, model building and for saving the model. I have kept all imports at the top for easy reference.

Read the dataset into a pandas dataframe

The loan default data is read into a pandas dataframe for analysis. Although the data was saved in a CSV file format, it is of Tab-Delimited form meaning that each data attribute is separated by a tab space. The original dataframe is stored in case I need to return to it later after making some intermediate transformations. There is an unnamed attribute in the dataset which represents the index. This attribute is dropped

Explore data analysis

I used the sweetviz module to perform exploratory data analysis. It performs extensive exploratory data analysis (central tendency, spread, bar charts, missing data analysis, correlation) on the entire dataframe and exports the results of the analysis as an interactive HTML-5 file. A static version is stored as well. If it shows "Error displaying widget" on the output of this ipynb, please ignore it, the HTML file will pop up as a new tab on your browser when it is done running. The column names are too long do be displayed legibly in the sweetviz correlation matrix. Thus, I have written a script to reduce the names to just the last part (after the 4th dot) for the relevant columns. I have added a numerical suffix in the cases where a name is duplicated.

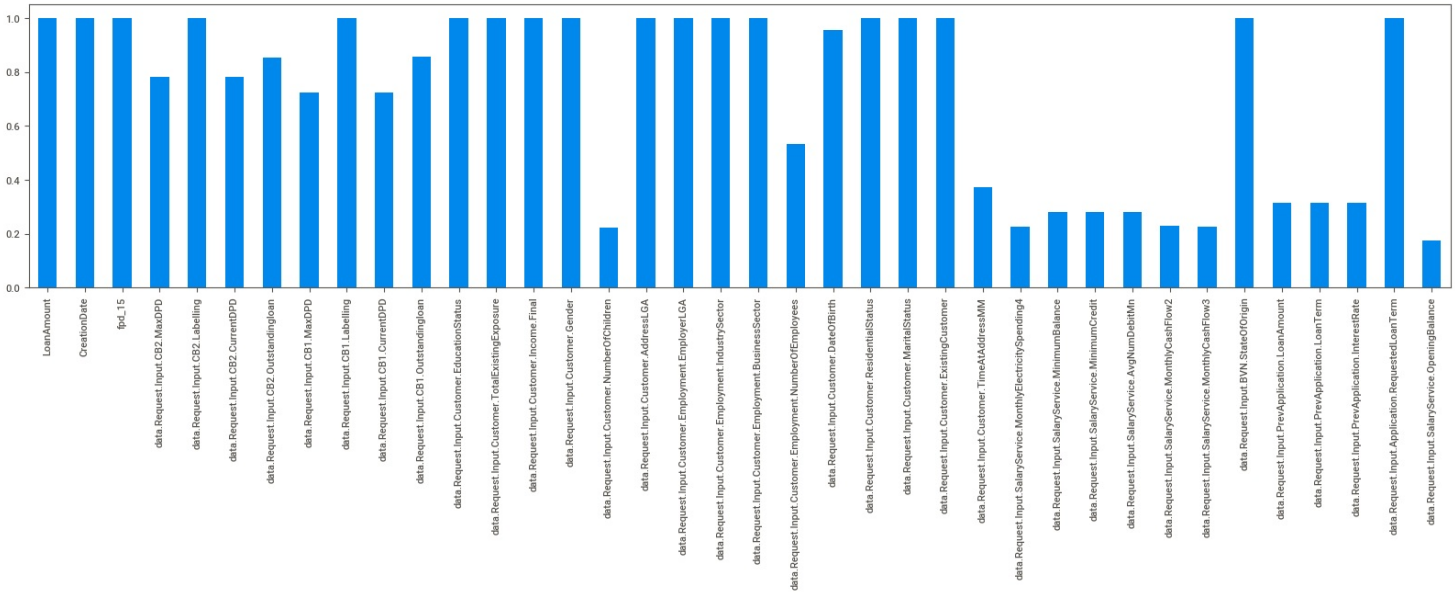
```
Index(['LoanAmount', 'CreationDate', 'fpd_15', 'MaxDPD', 'Labelling',
       'CurrentDPD', 'Outstandingloan', 'MaxDPD_2', 'Labelling_2',
       'CurrentDPD_2', 'Outstandingloan_2', 'EducationStatus',
       'TotalExistingExposure', 'Final', 'Gender', 'NumberOfChildren',
       'AddressLGA', 'EmployerLGA', 'IndustrySector', 'BusinessSector',
       'NumberOfEmployees', 'DateOfBirth', 'ResidentialStatus',
       'MaritalStatus', 'ExistingCustomer', 'TimeAtAddressMM',
       'MonthlyElectricitySpending4', 'MinimumBalance', 'MinimumCredit',
       'AvgNumDebitMn', 'MonthlyCashFlow2', 'MonthlyCashFlow3',
       'StateOfOrigin', 'LoanAmount_2', 'LoanTerm', 'InterestRate',
       'RequestedLoanTerm', 'OpeningBalance'],
      dtype='object')
                                           |          | [  0%]   00:00 -> (? left)
Report eda/report.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not pop up, regardless, the report IS sa
ved in your notebook/colab files.
```

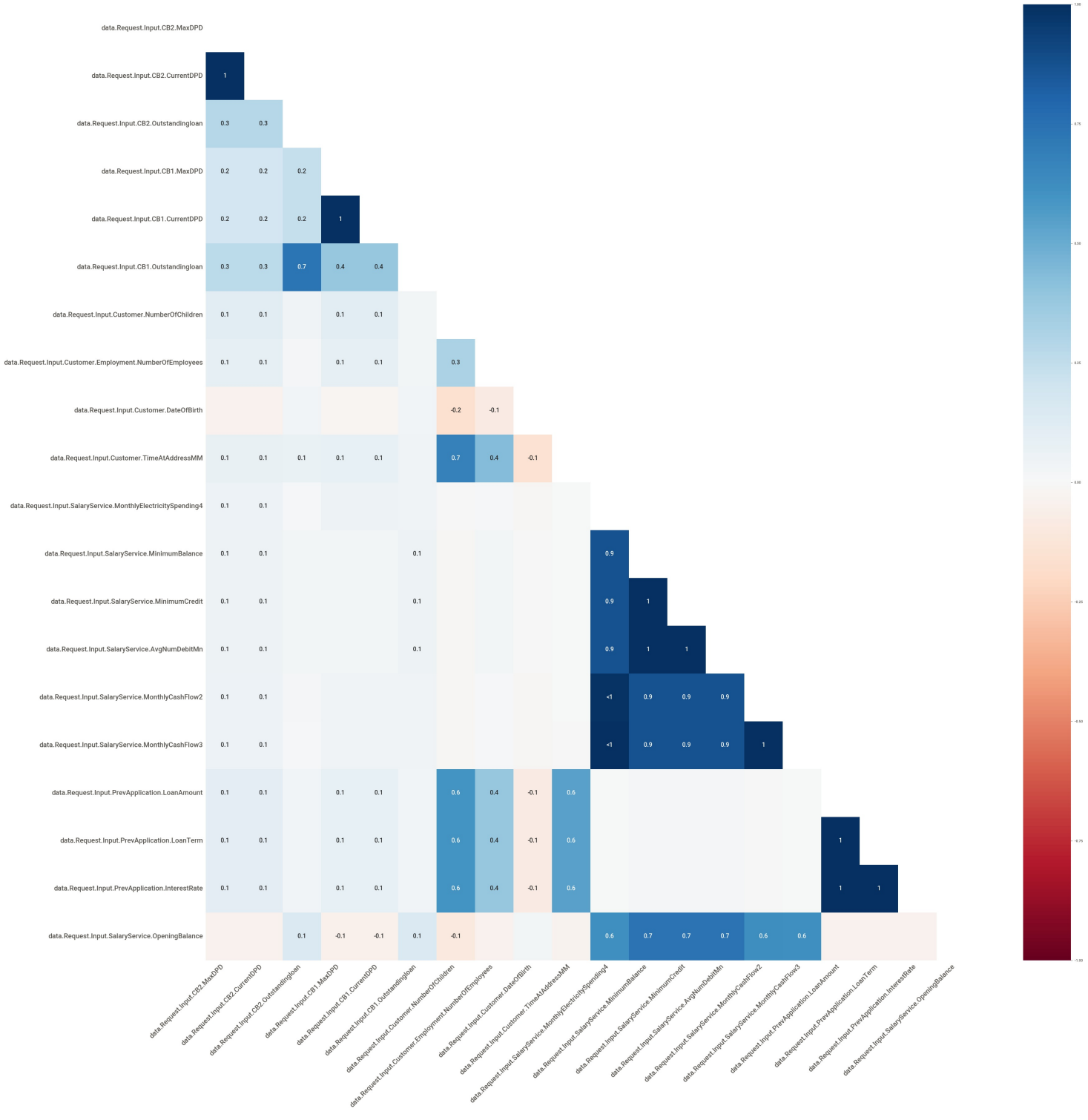
Analyse missing values

I restored the original names of the columns using the original verison of the dataframe. The dataset is then analysed for missing values. The percentage of missing values for each column is plotted a bar chart along with a heatmap of the correlation between missing values of the different attributes. Observations include:

1. About half of the attributes have missing data
2. The range of percentage missing is between 4% (date of birth) and 82% (opening balance)
3. Missing data rows are highly correlated for data from the same source e.g. CB1, CB2, SalaryService, PrevApplication

```
data.Request.Input.SalaryService.OpeningBalance                82.573875
data.Request.Input.Customer.NumberOfChildren                   77.768572
data.Request.Input.SalaryService.MonthlyElectricitySpending4   77.326705
data.Request.Input.SalaryService.MonthlyCashFlow3              77.299089
data.Request.Input.SalaryService.MonthlyCashFlow2              77.188622
data.Request.Input.SalaryService.MinimumBalance                71.969069
data.Request.Input.SalaryService.MinimumCredit                 71.969069
data.Request.Input.SalaryService.AvgNumDebitMn                 71.969069
data.Request.Input.PrevApplication.LoanAmount                  68.378901
data.Request.Input.PrevApplication.LoanTerm                    68.378901
data.Request.Input.PrevApplication.InterestRate                68.378901
data.Request.Input.Customer.TimeAtAddressMM                    62.607015
data.Request.Input.Customer.Employment.NumberOfEmployees       46.837890
data.Request.Input.CB1.CurrentDPD                              27.644297
data.Request.Input.CB1.MaxDPD                                  27.644297
data.Request.Input.CB2.CurrentDPD                              21.679094
data.Request.Input.CB2.MaxDPD                                  21.679094
data.Request.Input.CB2.Outstandingloan                         14.747307
data.Request.Input.CB1.Outstandingloan                         14.443524
data.Request.Input.Customer.DateOfBirth                         4.335819
data.Request.Input.Customer.ExistingCustomer                    0.000000
data.Request.Input.BVN.StateOfOrigin                            0.000000
data.Request.Input.Application.RequestedLoanTerm                0.000000
LoanAmount                                                      0.000000
data.Request.Input.Customer.MaritalStatus                       0.000000
data.Request.Input.Customer.ResidentialStatus                   0.000000
CreationDate                                                    0.000000
data.Request.Input.Customer.Employment.IndustrySector           0.000000
data.Request.Input.Customer.Employment.EmployerLGA              0.000000
data.Request.Input.Customer.AddressLGA                          0.000000
data.Request.Input.Customer.Gender                              0.000000
data.Request.Input.Customer.Income.Final                        0.000000
data.Request.Input.Customer.TotalExistingExposure               0.000000
data.Request.Input.Customer.EducationStatus                     0.000000
data.Request.Input.CB1.Labelling                                0.000000
data.Request.Input.CB2.Labelling                                0.000000
fpd_15                                                          0.000000
data.Request.Input.Customer.Employment.BusinessSector           0.000000
dtype: float64
```





Handle missing data

Credit Bureau data

The credit bureau data seems to have a missing data rate of between 14% and 27%. Because the credit bureau data comes from two sources (CB1 and CB2) that represent the same information (customer credit history), I have opted to merge them by taking the maximum value between the data provided by each credit bureau. This dropped the credit bureau missing data rate from to between 9% and 11%.

Salary data and Previous application data

The data from the bank statement (SalaryService) and from previous loan application (PrevApplication) have a very high missing data rate (between 68% and 82%). Because of the high rates, it might not be wise to fill in this data by imputation (replacing the missing values with their average value or 0). Imputation may introduce an unwanted bias as it would imply that more than half of that attribute is a result of modelling rather than observation. Ideally, if the reason for the missing data was known, an appropriate imputation strategy could be formulated. Instead, I have opted to drop these attributes from the dataset. I have specifically dropped columns with missing rates more than 30%. But before dropping these columns, I have created indicator columns (columns that indicate if another column has a missing value for an observation) which could be informative to the loan default problem. Because the missing data from 'PrevApplication' fields are highly correlated (> 90%), I have used 'PrevApplication.LoanAmount' as a proxy for these fields when creating an indicator column. Similarly, I have used the 'SalaryService.MinimumBalance' field as a proxy in the case of the 'SalaryService' fields.

The new dataset now has a missing data rate of between 4% and 11%. In a later code block, I will perform imputation on these.

```
data.Request.Input.CB.Outstandingloan                    11.267606
data.Request.Input.CB.CurrentDPD                          9.389671
data.Request.Input.CB.MaxDPD                              9.389671
data.Request.Input.Customer.DateOfBirth                   4.335819
CreationDate                                              0.000000
data.Request.Input.Customer.Employment.IndustrySector     0.000000
previousApplication                                       0.000000
fpd_15                                                    0.000000
data.Request.Input.Customer.TotalExistingExposure         0.000000
data.Request.Input.Customer.ResidentialStatus             0.000000
data.Request.Input.Customer.MaritalStatus                 0.000000
data.Request.Input.Customer.Income.Final                  0.000000
data.Request.Input.Customer.Gender                        0.000000
data.Request.Input.Customer.ExistingCustomer              0.000000
data.Request.Input.Customer.Employment.BusinessSector     0.000000
data.Request.Input.Customer.Employment.EmployerLGA        0.000000
LoanAmount                                                0.000000
data.Request.Input.Customer.EducationStatus               0.000000
data.Request.Input.Customer.AddressLGA                    0.000000
data.Request.Input.CB2.Labelling                          0.000000
data.Request.Input.CB1.Labelling                          0.000000
data.Request.Input.BVN.StateOfOrigin                      0.000000
data.Request.Input.Application.RequestedLoanTerm          0.000000
salaryService                                             0.000000
dtype: float64
```

Preprocessing

Handling date features

There are two date features in the dataset, the creation date and the date of birth. Because date features cannot be utilised by an ML model as is, I have decided to convert them to numerical values that represent the age of the loan and the customer age in months. Sometimes experiments and lapses in loan decisoning may occur over time period which then skews default rate. Having the loan age in months makes it easier for a model to capture this possibility.

Assumptions:

1. I have assumed that the 'CreationDate' attribute refers to the date in which the loan was granted.
2. I have assumed that the dataset is a snapshot of the loan defaults/non-defauls at a particular point in time since they were taken

Data cleaning: uniform format for state of origin

I noticed that the state of origin column has 47 distinct values (46 states and 1 'other'). This is much more than the 37 expected (36 states of Nigeria and 1 'other'). Upon investigation, I noticed that some of the states e.g. lagos appears twice: once in lowercase and again in uppercase. I decided to convert these values into lower case. This reduced the unique count to 37 as expected.

```
no. of states before cleaning: 45
no. of states after cleaning: 37
```

Set Fetaures and Target attributes

The target column 'fpd_15' is written to a seperate pandas Series and the feature columns are separated into numerical and categorical for further processing

Assumptions:

1. I have assumed that fpd_15 refers to first payment default 15 days past due i.e. indicative of loan default. THis is because, of all the attributes, it is the closest in meaning to loan default.

	LoanAmount	data.Request.Input.Application.RequestedLoanTerm	data.Request.Input.BVN.StateOfOrigin	data.Request.Input.CB.CurrentDPD	da
0	1350000.0	18.0	kaduna state	0.0	
1	670000.0	17.0	abia state	0.0	
2	5410000.0	16.0	taraba state	0.0	
3	110000.0	14.0	jigawa state	0.0	
4	180000.0	10.0	ogun state	0.0	

5 rows × 23 columns

Preprocessing: Imputation

To account for the missing rate of 4% - 11% on the remaining numerical attributes, I have used mean imputation. This should not be too troublesome as it is only being done on a very small portion of the data (three columns and a maximum of 11% of observations). It will prevent information loss caused by discarding entire observations with missing data for these attributes

	LoanAmount	data.Request.Input.Application.RequestedLoanTerm	data.Request.Input.CB.CurrentDPD	data.Request.Input.CB.MaxDPD	data.Req
0	1350000.0	18.0	0.0	0.0	
1	670000.0	17.0	0.0	0.0	
2	5410000.0	16.0	0.0	0.0	
3	110000.0	14.0	0.0	0.0	
4	180000.0	10.0	0.0	0.0	

Data Cleaning: Outlier Removal

I noticed that the there are also outliers in the data provided. For example, 1)the minumum income value is a negative figure, 2)the max days past due is equivalent to more than 100 years, 3)the maximum income value is in the order of a trillion. These are likely to be data capture errors. I have used Local Outlier Factor, a density based outlier detection algorithm to remove such entries. At a contamination level of 1% (i.e. amount allowed considered outliers), it is able to leave reasonable values for the maximum and minimum values of the numerical features without losing too many observations.

```
MAX AND MIN VALUES BEFORE OUTLIER REMOVAL:
                                                        Max          Min
LoanAmount                                        6.000000e+06      80000.0
data.Request.Input.Application.RequestedLoanTerm  4.300000e+01          2.0
data.Request.Input.CB.CurrentDPD                  4.060000e+03          0.0
data.Request.Input.CB.MaxDPD                      4.474200e+04          0.0
data.Request.Input.CB.Outstandingloan             5.686400e+08          0.0
data.Request.Input.Customer.ExistingCustomer      1.000000e+00          0.0
data.Request.Input.Customer.Gender                1.000000e+00          0.0
data.Request.Input.Customer.Income.Final          6.317924e+12 -189469000.0
data.Request.Input.Customer.TotalExistingExposure 5.992000e+06      82000.0
previousApplication                               1.000000e+00          0.0
salaryService                                     1.000000e+00          0.0
loanAge                                           1.800000e+01          8.0
data.Request.Input.Customer.Age                   7.900000e+02        268.0
no. of original observations:  3621


MAX AND MIN VALUES AFTER OUTLIER REMOVAL:
                                                       Max      Min
LoanAmount                                       6000000.0  80000.0
data.Request.Input.Application.RequestedLoanTerm        43.0      2.0
data.Request.Input.CB.CurrentDPD                      4060.0      0.0
data.Request.Input.CB.MaxDPD                         44742.0      0.0
data.Request.Input.CB.Outstandingloan             30250000.0      0.0
data.Request.Input.Customer.ExistingCustomer             1.0      0.0
data.Request.Input.Customer.Gender                       1.0      0.0
data.Request.Input.Customer.Income.Final         569519000.0      0.0
data.Request.Input.Customer.TotalExistingExposure   5992000.0  82000.0
previousApplication                                      1.0      0.0
salaryService                                            1.0      0.0
loanAge                                                 18.0      8.0
data.Request.Input.Customer.Age                        790.0    268.0
no. of remaining observations:  3584
```

Split data into training and testing sets

To facilitate machine learning, the data has been split into training and testing sets at an 80:20 ratio. I have opted not to further split the training set into core triaining and validation sets because I will be using k-fold Cross Validation later on.

```
no. of training data:  2867
no. of training data:  717
```

Preprocessing: Target encoding

Categorical features cannot be directly utilised by machine learning models. Because of this, I have opted to encode these fetaures in to a numerical format. For this purpose, I have opted for target encoding which replaces the feature values with the percentage of positive class (defaulters) that match that feature value. It is fitted only on the training set and then used to transform both the training and test sets. This is in order to prevent information leakage from the test set into the training set. Alternative methods include: frequency encoding, and one-hot encoding. I have opted for target encoding because if the dataset is homogeneous (which it is assumed to be as training and test sets are randomly picked), it captures a lot of useful information in informing model formation. It also does not create a overly wide dataset which one-hot encoding tends to do on high cardinality features.

Preporcessing: Class Imbalance

The number of non-defaulters is 24% more than the number of defaulters in the dataset. This introduces a class imbalance which causes a bias towards the majority class in model building. In order to prevent this bias, I have opted to balance the classes by random oversampling. Alternative methods include underdampling, and SMOTE (a model based method). I have opted for random oversampling because it retains all samples from the majority class (against undersampling) and does not rely on a model which may introduce its own bias to the dataset (against SMOTE).

```
BEFORE OVERSAMPLING
no. of defaulters:  1279
no. of non-defaulters:  1588
percentage of defaulters in dataset:  44.61109173351936 %

AFTER OVERSAMPLING
no. of defaulters:  1588
no. of non-defaulters:  1588
percentage of defaulters in dataset:  50.0 %
```

Preprocessing: Imputation for categorical features

Now that the categorical features have been target encoded into numerical values, I perform mean imputation in case of any missing values in these features similarly to what was done for the numerical variables. This is a failsafe and perhaps redundant step as these shouldn't have any missing values if they were successfully target encoded.

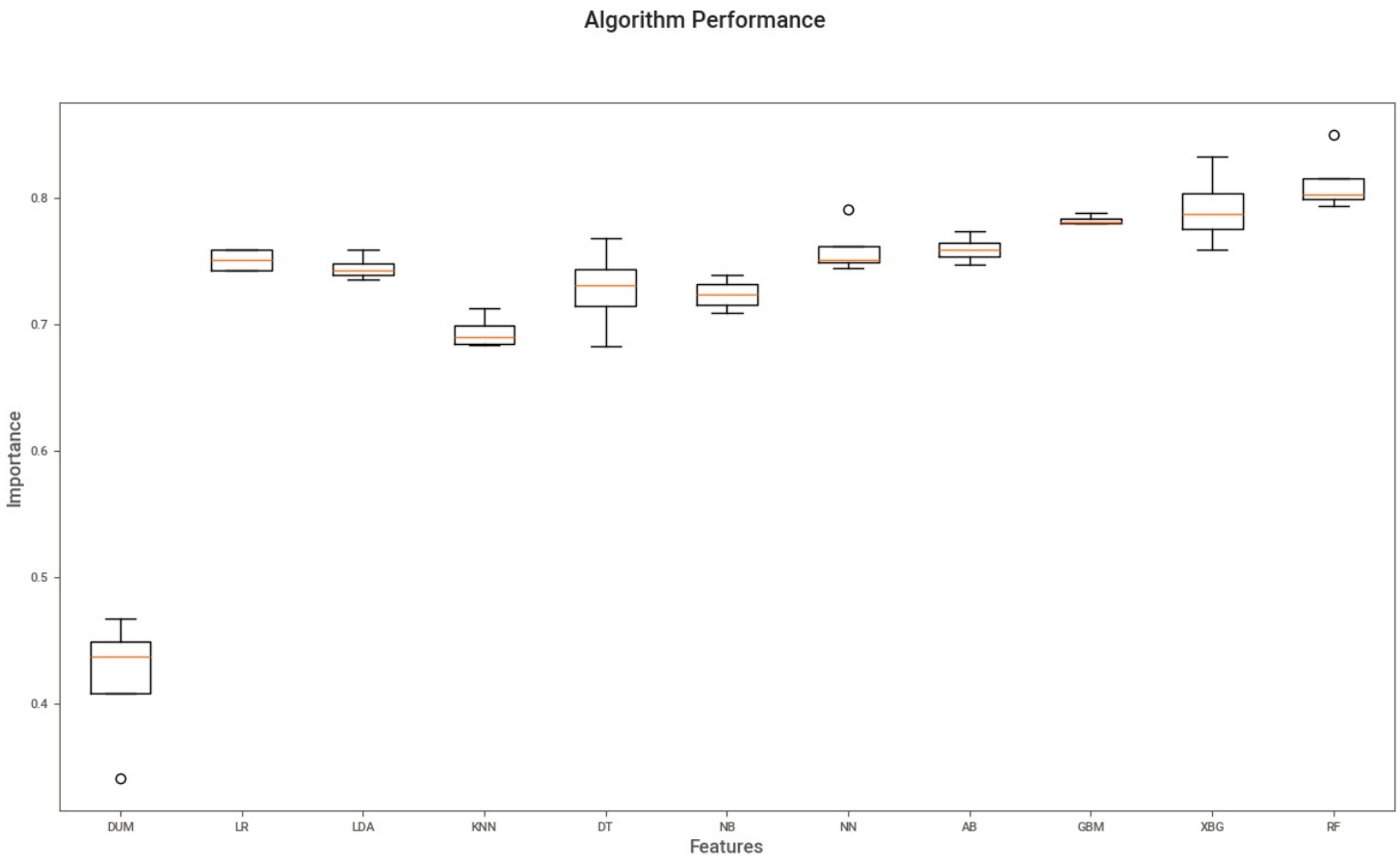
Preprocessing: Standardisation

In this step, I standardise the data. Standardisation involves scaling the feature values to zero mean and unit variance. This is done so that the scale of each feature is similar. It has the effect of improving certain models like distance based models (K-Nearest Neighbours) and regression based models (Logistic Regression). Because these models are candidates for selection, standardisation will help boost their performance. The scaler is fitted only on the tarining data and this fit is used to transform both the training data and the test data. I have restored the binary features to their original form because standardisation is not needed in these cases and may actually cause information loss in some models.

Model Building: Model selection

In this next step, I train different classes of machine learning models (based on different algorithms) on their default values to determine the best type to use for the analysis. Among these models, I include a random guess dummy model as a baseline to determine if there is any predictability from the feature set. I use 4-fold cross validation (4-fold so that the validation sets and the test set are of the same size 20% = 25% of 80%). I compare the mean CV score, the standard deviation, and ratio between them to determine which model is best to move forward with. In this case, I am using the accuracy as the metric to optimise because the dataset is balanced and there is no disclosed trade-off advantage between the positive and negative classes. All models far outperform the dummy model, indicationg that the feature set has predictability. The Random Forest Model shows the most promise with a the highest mean CV score and a low standard deviation.

```
DUM: score - 0.420655 stdev - (0.047651) ratio- 8.827766
LR: score - 0.751259 stdev - (0.008186) ratio- 91.769231
LDA: score - 0.744962 stdev - (0.008928) ratio- 83.442386
KNN: score - 0.693955 stdev - (0.011577) ratio- 59.940898
DT: score - 0.727960 stdev - (0.030592) ratio- 23.795883
NB: score - 0.723866 stdev - (0.011418) ratio- 63.398078
NN: score - 0.759446 stdev - (0.018403) ratio- 41.268470
AB: score - 0.759446 stdev - (0.009633) ratio- 78.838686
GBM: score - 0.782431 stdev - (0.003604) ratio- 217.115458
XBG: score - 0.791562 stdev - (0.026576) ratio- 29.785392
RF: score - 0.812028 stdev - (0.022306) ratio- 36.403467
```

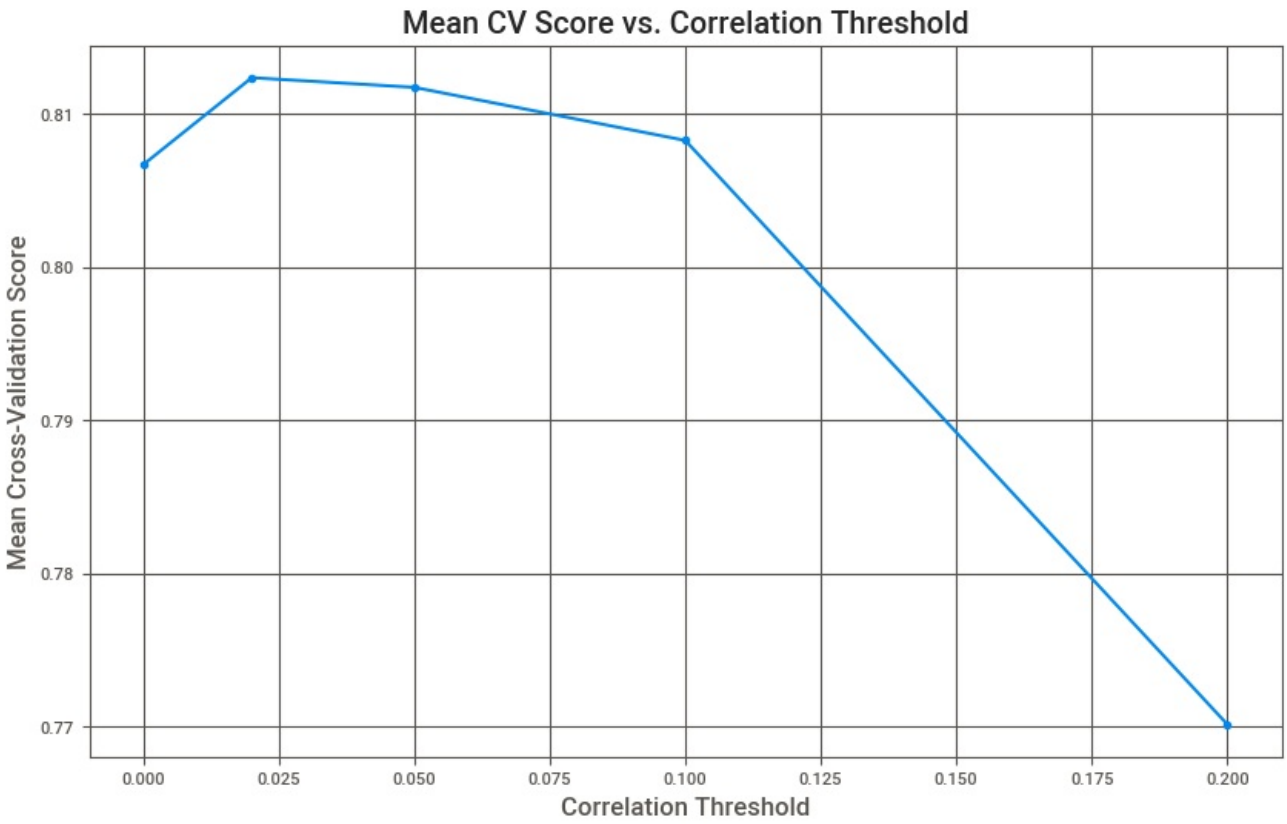


Feature Selection: Correlation screening cross validation

The dataset has many features relative to the number of observations. This may lead to overfitting which affects model performance out of sample.

Some features may contribute more noise than signal to the machine learning model. In order to reduce this, one method is correlation screening. In this case, I screen the features to eliminate those that are least correlated with the target variable. In order not to eliminate too many features, I built and cross-validated a model for a set of correlation thresholds and compared the results. Screening out features with less than 5% correlation resulted in the best model. Although the difference in performance is small compared to using all the features, removing unneccessary features is in line with with the principle of Occam's razor and helps achieve a higher level of model parsimony (robustness).

```
Threshold: 0.0
Number of features: 23
Mean CV Score: 0.8066750629722921


Threshold: 0.02
Number of features: 20
Mean CV Score: 0.8123425692695214


Threshold: 0.05
Number of features: 18
Mean CV Score: 0.8117128463476071


Threshold: 0.1
Number of features: 13
Mean CV Score: 0.8082493702770781


Threshold: 0.2
Number of features: 3
Mean CV Score: 0.7701511335012594
```



Feature Selection: correlation with target

Following up from the previous code block, I eliminate the features with less than 2% correlation with the target. This leaves 20 features for model building. Before doing so, I save the current feature set for future reference.

```
the number of features before correlation screening:  23
the number of features after correlation screening:  20

the features that were removed:  ['data.Request.Input.CB1.Labelling', 'data.Request.Input.Customer.EducationStatus', 'da
ta.Request.Input.Customer.MaritalStatus']
```

Feature Selection: correlation between features

Correlation screening can also be used to screen out features that are higly correlated with each other. In some models this may cause multi-collinearity. Random forest can be robust to this but it is better avoided to reduce redundancy and also for accurately assessing feature importance. In this case, the previously created PreviousApplication indicator column is highly correlated with the existing customer column. In fact, they carry the exact same information. Thus, it is screened out.

```
identified columns:  ['previousApplication']
the number of features after correlation screening:  19
```

Hyperparameter Optimisation

In order to arrive at the best model on the training set (logically, this should also extend to the test set), I perform some hyperparameter optimisation. More specifically, I perform 4-fold cross validation accross a number of different options for the the diffrerent hyperparameters that are relevant to a Random Forest Classifier. I then chose the set of hyperparameters that yield the best model. Because, there are many hyperparameters, and each combination of hyperparamter settings will undergo 4-fold CV, the number of models trained becomes exponential. This process takes a lot of time. Instead, to limit the options for each hyperparameter (the numerical ones in particular), I opted to test a range of values for each hyperparamter separately while leaving the rest at their default values. In deciding which hyperparamters to select, I opted for the set that not only displayed high performance but also low variation between the training and test sets. This is in order to prevent overfitting to the training set.

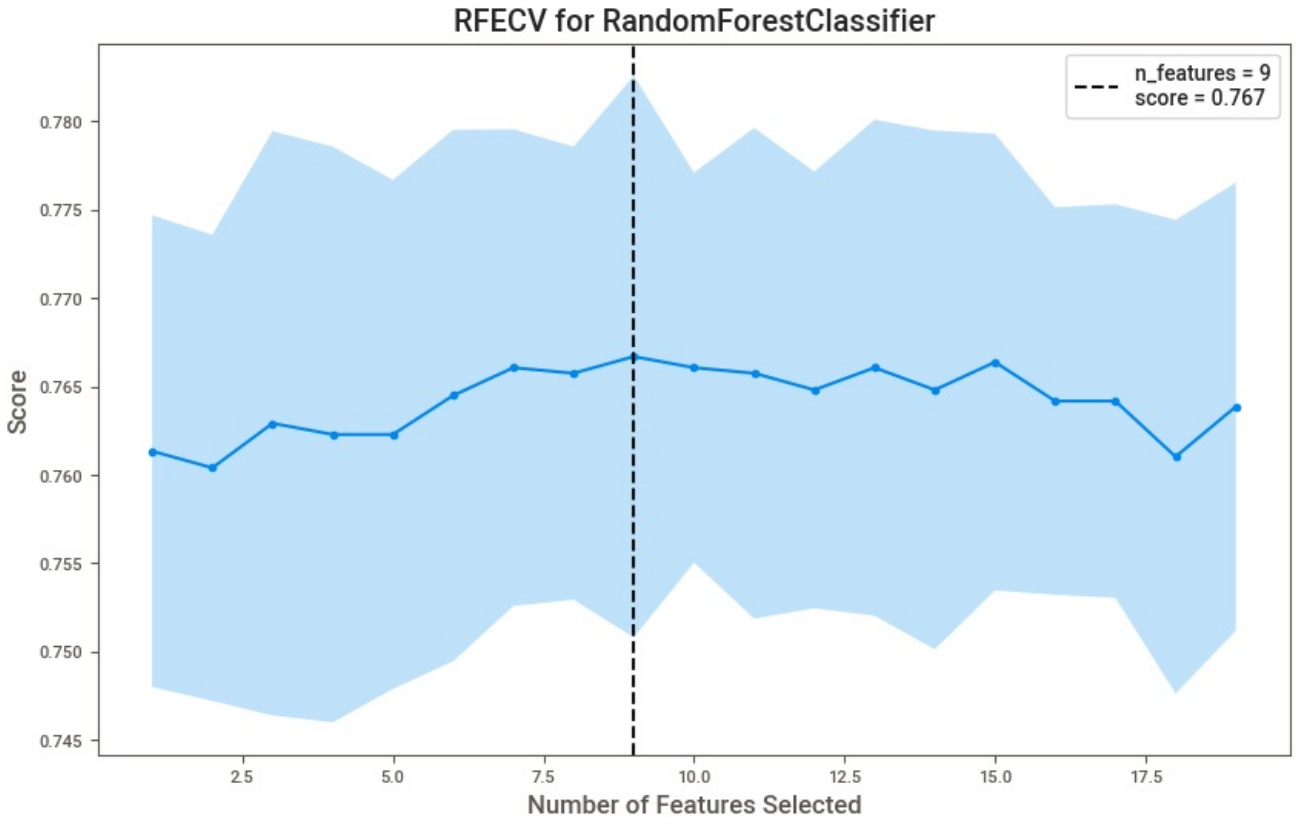
```
Fitting 4 folds for each of 1 candidates, totalling 4 fits
```

```
#1 Test: 0.756297 (0.009937) Train: 0.776029 (0.002663) with: {'bootstrap': True, 'class_weight': {0: 1, 1: 1.1}, 'crite
rion': 'entropy', 'max_depth': 4, 'max_features': 2, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 80}
and train-test variation: 0.019731
```

```
Best: 0.756297 using {'bootstrap': True, 'class_weight': {0: 1, 1: 1.1}, 'criterion': 'entropy', 'max_depth': 4, 'max_fe
atures': 2, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 80}
```

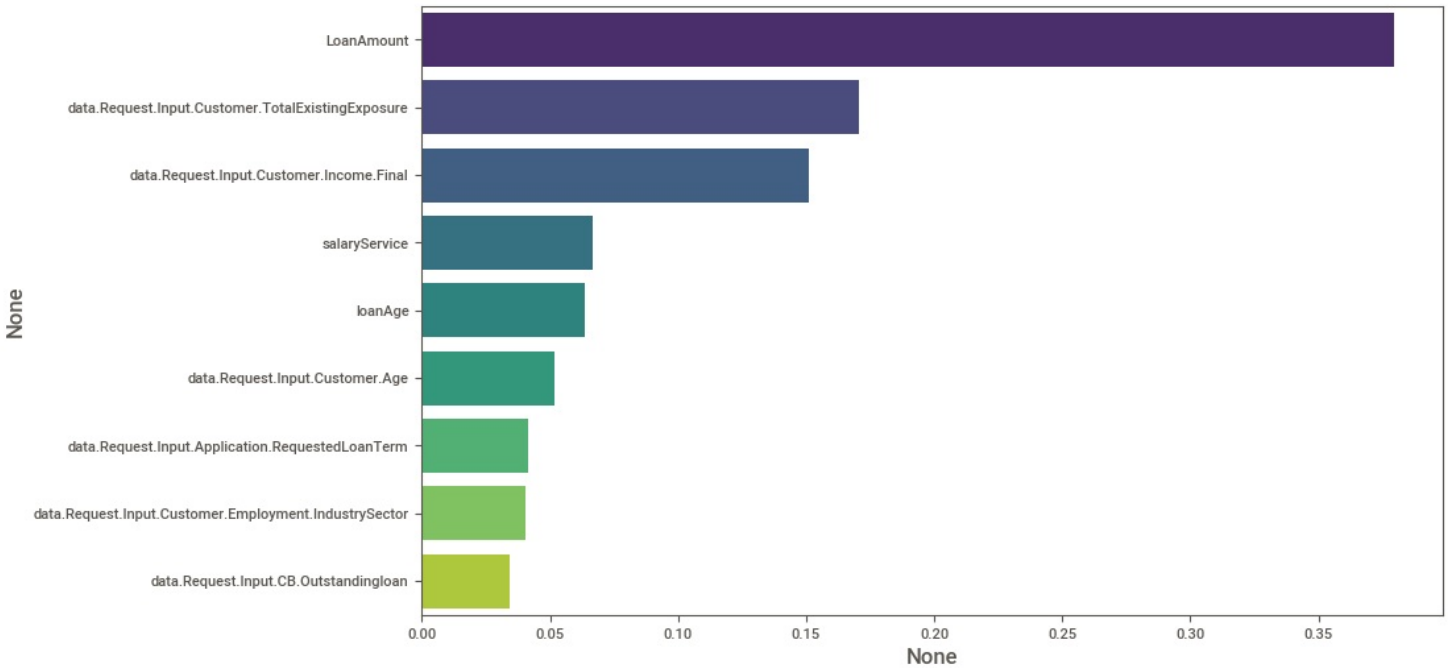
Feature Selection: Recursive Feature Elimination

Now that the best hyperparameters have been selected, the next step is to return to feature selection in order to eliminate any redundant features with the current hyperparamter selection. Recursive Feature Elimination Cross Validation (RFE) is a method which starts by perfoming cross validation on the model using all the features presented and then uses the calculated feature importances to eliminate the least contribting feature. It repeats this process iteratively until only a single feature remains. Then, the cross validation scores can be compared accross the different fetaure selection options. The trend shows an increase in performance as no. of features increased, followed by a flattening and then a slight decrease. In this case, it shows that the optimal number of features is 9. Although similar results were achieved by including more features, adding these features without increasing the amount of data available will likely lead to overfitting.



Feature Importance

In this step, I assess the feature importance of the model that has been built. First I use RFE to select the most important features (In this case, 9 of the features left are considered relevant). Then I determine and plot the feature importances on a bar chart.



Test the model on the test set

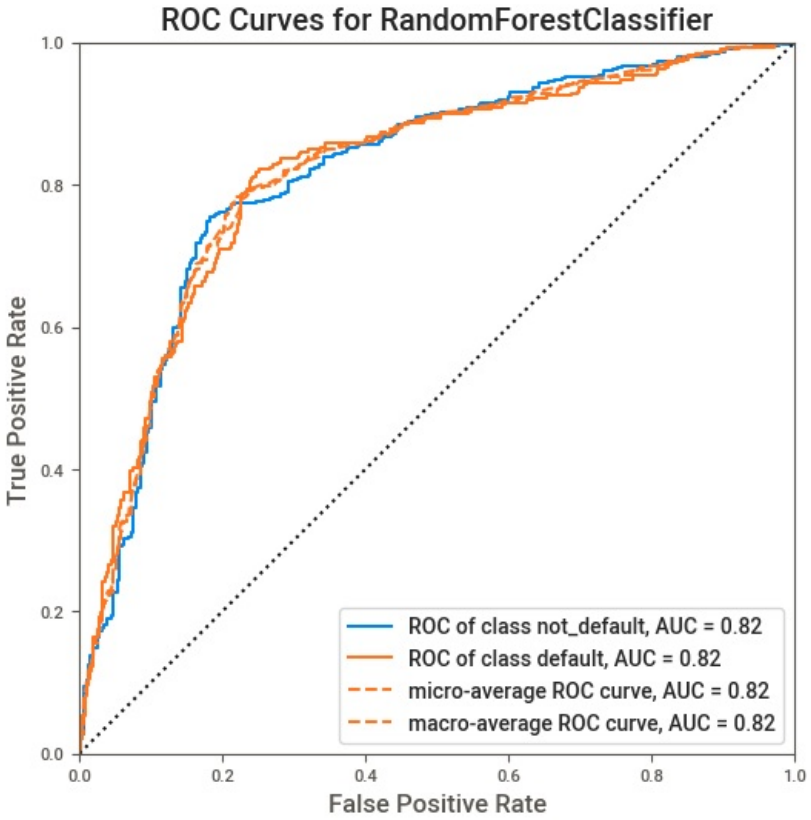
Now that the model building exercise is concluded. I proceed to test the model on the test set. This will determine if the model has been overfit/underfit during the training process and will assess its readiness to go into production. The test set has an accuracy score of 78%. This is not too far off from the 77% average on the CV test set and equal to 78% average on the CV train set. Thus, the model generalises well. The results also have a relatively well balanced confusion matrix with a slight tendency to misclassify some non-defaults as defaults, which may be a preferable trade-off in the context of loan default prediction.

```
confusion matrix:
 [[303  95]
 [ 63 256]]

classification report:
              precision    recall  f1-score   support

           0       0.83      0.76      0.79       398
           1       0.73      0.80      0.76       319

    accuracy                           0.78       717
   macro avg       0.78      0.78      0.78       717
weighted avg       0.78      0.78      0.78       717

Accuracy:  0.7796373779637378
ROC AUC score:  0.7819071848269561
F1-score:  0.764179104477612
Recall:  0.8025078369905956
Precision:  0.7293447293447294
```

Evaluation: ROC Curve

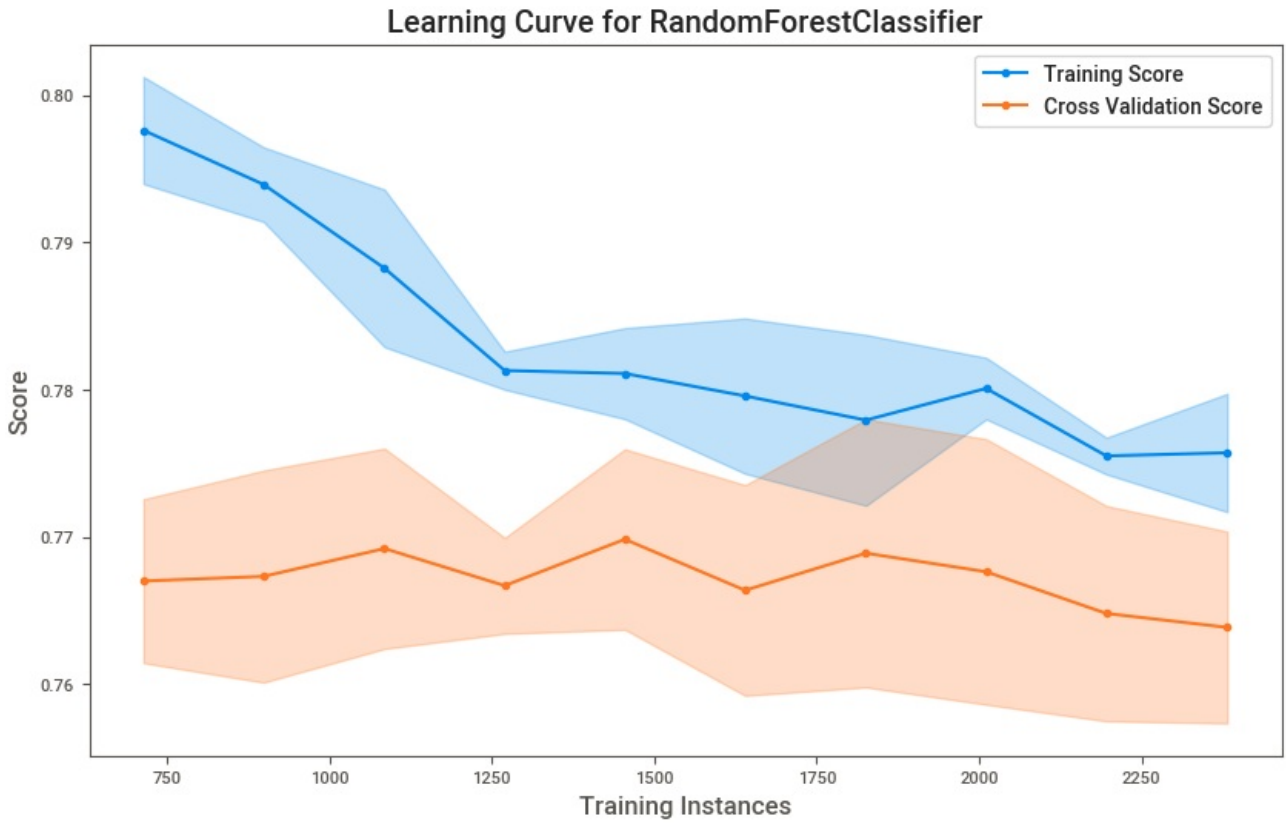
The Receiver-Operator Characteritic (ROC) curve is used to discern how well the model distinguishes between defaulters and non-defaulters. It is a plot of the True positive rate against the false positive rate. The AUC is the area covered by this graph and the gini coefficient is the area between the curve and the digonal. From this curve, the following can be noted:

1. The model significantly outperforms random guessing (AUC/ gini coefficient)
2. It can be visually discerned that the model performs as well on the positive class as on the negative class as their curves are almost identical (ROC curves)
3. The model has an identical trade-off between the true positive rate and the false positive rate (bottom-left compared to top right)



Evaluation: Learning Curve

In order to discern the stability of the model with respect to data availability, the learning curve can be utilised. This shows how the model accuracy changes as more data is added to the **training set** using k-fold cross validation. The learning curve shows that the model overfits the training data when a small amount of data is utilised but as additional data is added its performance is more generalisable to the test data. The learning curve also shows that the CV score is flat even as more data is utilised. This indicates that integrating additional training data will likely not boost the performance of the model out off sample. This is likely because of the feature pruning that was performed. Incorporating additional features may improve performance when additional data is added to the dataset but without additional data it will most likely overfit the training data and not generalise to the test data.



Export artifacts

Not that the model building exercise is concluded, I proceed to export the artifacts that were generated in the process. These include the random forest model, the standard scaler, the dictionary of target encoders, and the list of selected features.

Deployement

The exported model can be deplyed on an endpoint. I use the Flask module in python to deploy to host the model on a local server. The model is made available via the 'predict' endpoint. The user simply sends a JSON payload to the endpoint and gets the prediction and the model will generate a prediction and the probability of default. The codeblock below creates the python file that runs the deployment.

```
Overwriting app.py
```

Run the deployment

To run the deployment, install python and copy the code below and paste in command prompt (windows) or Terminal (Mac/Linux). It should create a local deployment on your network using your PC as a server

Test the deployment

The script below tests the deployment. It reads a random row from the original dataset and performs the same data cleaning, pre-processing, and feature engineering that was used in building the model. It then calls the model's endpoint with the processed features. The endpoint should respond with a JSON containing the prediction and the probability of default. Uncomment the code to run it after running "python app.py" on Terminal or Command prompt.

```
The selected observation is:

{"LoanAmount":130000.0,"CreationDate":"2023-04-28","fpd_15":0,"data.Request.Input.CB2.MaxDPD":0.0,"data.Request.Input.CB
2.Labelling":"NTC","data.Request.Input.CB2.CurrentDPD":0.0,"data.Request.Input.CB2.Outstandingloan":0.0,"data.Request.In
put.CB1.MaxDPD":0.0,"data.Request.Input.CB1.Labelling":"Prime","data.Request.Input.CB1.CurrentDPD":0.0,"data.Request.Inp
ut.CB1.Outstandingloan":0.0,"data.Request.Input.Customer.EducationStatus":"Secondary","data.Request.Input.Customer.Total
ExistingExposure":439000.0,"data.Request.Input.Customer.Income.Final":2914000.0,"data.Request.Input.Customer.Gender":1,"
data.Request.Input.Customer.NumberOfChildren":1.0,"data.Request.Input.Customer.AddressLGA":"Epe","data.Request.Input.Cus
tomer.Employment.EmployerLGA":"Badagry","data.Request.Input.Customer.Employment.IndustrySector":"Motion picture, sound r
ecording and music publishing activities","data.Request.Input.Customer.Employment.BusinessSector":"Extraterritorial orga
nizations","data.Request.Input.Customer.Employment.NumberOfEmployees":3.0,"data.Request.Input.Customer.DateOfBirth":"199
1-04-19","data.Request.Input.Customer.ResidentialStatus":"Owner","data.Request.Input.Customer.MaritalStatus":"Divorced",
"data.Request.Input.Customer.ExistingCustomer":1,"data.Request.Input.Customer.TimeAtAddressMM":89.0,"data.Request.Input.
SalaryService.MonthlyElectricitySpending4":null,"data.Request.Input.SalaryService.MinimumBalance":null,"data.Request.Inp
ut.SalaryService.MinimumCredit":null,"data.Request.Input.SalaryService.AvgNumDebitMn":null,"data.Request.Input.SalarySer
vice.MonthlyCashFlow2":null,"data.Request.Input.SalaryService.MonthlyCashFlow3":null,"data.Request.Input.BVN.StateOfOrig
in":"Katsina State","data.Request.Input.PrevApplication.LoanAmount":28000.0,"data.Request.Input.PrevApplication.LoanTerm
":1.0,"data.Request.Input.PrevApplication.InterestRate":30.0,"data.Request.Input.Application.RequestedLoanTerm":13.0,"da
ta.Request.Input.SalaryService.OpeningBalance":null,"data.Request.Input.CB.CurrentDPD":0.0,"data.Request.Input.CB.MaxDPD
":0.0}


the prediction is:

{'prediction': '1: default', 'probability of default': 0.7684742491722146}
```