Home Credit : Machine Learning

After the data wrangling, exploratory data analysis and statistical inference my wrangled data is **ALMOST** ready to be fed to the machine learning models.

Type of Problem

The problem here is a binary classification problem and my features here are a mix of continuous and categorical variables.

Feature Engineering

In the merged data set we now have new features included but we still have few steps to be executed as part of the feature engineering process and also make sure to avoid data leakage.

That's why the merged data set with new features is divided into train and test data set after performing the below steps in such a way that train data doesn't get influenced by the test data in any way (data leakage)

- Handling missing values
- Encoding categorical variables
- Scaling variables
- Changing data types of variables where required

Size of the training data set: # of rows – **246008**, # of columns – **364**

Size of the test data set: # of rows – **61503**, # of columns – **364**

Once we have the train and test data, we perform feature selection using the training data set.

Feature Selection

As there were 364 features in the combined data set, I ran a random forest model on the whole data set to find the most important features by rank so that only the most important features could be used as an input to the model, provided doing so doesn't compromise much with the model results. Doing so, would help in reducing the processing time of training the model and thus also saving CPU resources.

Why random forest model was chosen for feature selection

- Feature selection is performed to look at the possibility whether using a limited set of features (the most important ones) can give almost the same performance when compared to the scenario in which we use all the features. If there is such a possibility, then it will be desirable as this will let the model be faster and use less computing resources.

- The reason for which Random Forest was chosen as the model for feature selection is that while determining feature importance, it takes into consideration the effect of all the features at the same time rather than considering them individually. Not only this, it also automatically considers the interaction between feature variables.

Points to consider for the choice of models

- **Time constraint (Latency):** As the problem definition doesn't mention any time constraint on time taken to classify an applicant who is likely to default, I have assumed that choice of model is flexible in terms of time taken by the model to predict the output.

- **Model Interpretability:** I assume in this project that the prediction power of the model is lot more important than the model interpretability in terms of features. The significance of different features was covered in detail as part of the EDA.

- **Evaluation Metric:** The **AUC score** is chosen as the metric for evaluating the performance of the classification model as this metric checks the error rate for each threshold in contrast to misclassification rate which checks the same for only one threshold (0.5). The Business would most probably like to see the performance of the model at different thresholds so that they can decide their optimum loan providing strategy.

Choice of Model

- Depending on the points discussed in the previous slide I tried the below models from different families.

  Parametric Models: Logistic Regression

  Non-Parametric Models (Bagging): Random Forest

Non-Parametric Models (Boosting): XGBoost and Light GBM

Creating a sample dataset to pass to the GridSearchCV library for hyperparameter tuning

The combined data set with 246008 observations and 364 features was divided into **no sample** (99.9%) and **sample data** (0.1%).

The above division resulted into a sample feature data set of 24601 observations and 364 features variables and sample target data set of 24601 observations and one target variable (The Default Status).

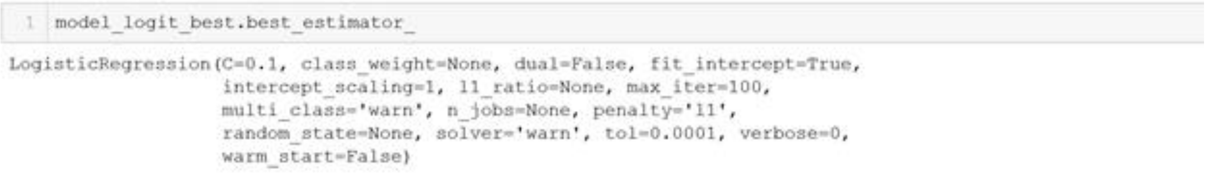
Running the Logistic Regression Model

The hyperparameter values to be passed to the final Logistic Regression model were determined by running a grid search with 5-fold cross validation with the below set of hyperparameters.

```
params_logit = {
                    'C': [0.001,0.01,0.1,1,10,100],
                    'penalty' : ['l1','l2']
                    }
model_logit = LogisticRegression()
model_logit_best = bestmodel(model_logit,params_logit)
```

To limit the usage of time for running the resource intensive grid search function the grid search function was fit using the sample training feature and sample training target datasets created in the previous step.

The below were the values of best parameters returned by the GridSearchCV function for the Logistic Regression model.



```
model_logit_best.best_estimator_
```

```
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l1',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

Training the model on the complete data set using the chosen hyperparameter values

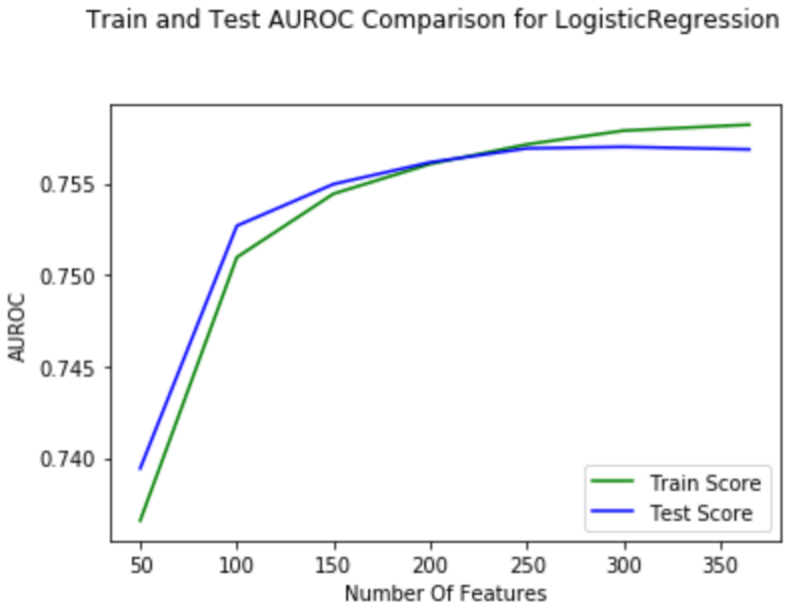
Once we got the best parameter values from grid search, the logistic regression model was again trained using the complete training data set (246008, 364).

- **Training the model on training data**

  The Logistic Regression model with the best parameters were retrained on the training data created above for different number of top ranked features w.r.t feature importance created during feature selection (Please refer to the **Feature Selection** section)

  modeldetails(model_logit_best)

- **Evaluating the performance of the trained model with respect to train and test data and also with respect to # of top features**

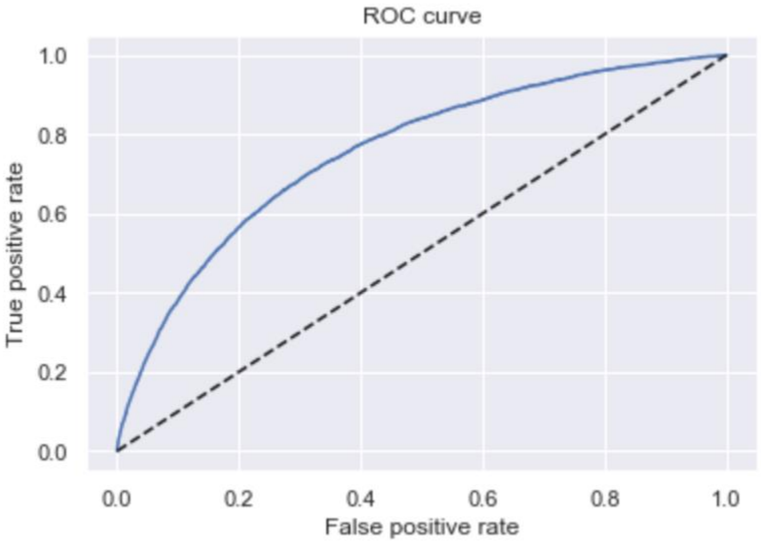


We can see that after 200 features, increasing # of features doesn’t result in substantial increase in AUROC score, hence top 200 features appears to be optimum # of features to be passed to the final Logistic Regression without compromising much on the model result.

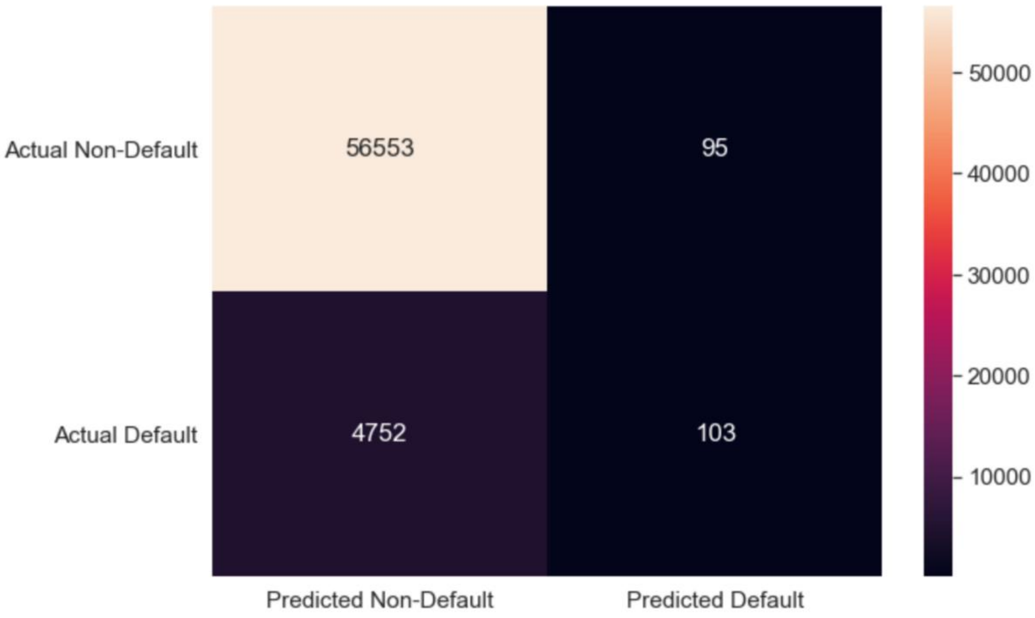
```
logit_finalmodel_matrix=

finalmodelperformance(200, model_logit_best.best_estimator_,0.5)
```

When the model was run on the unseen data it returned an AUC score of

AUC Score (Logistic Regression Model with 150 features): 0.7560



`Confusion Matrix (Logistic Regression Model with all 200 features)`



I chose the logistic regression model with 200 features along with the chosen hyperparameters as one of the contenders for the final model selection.

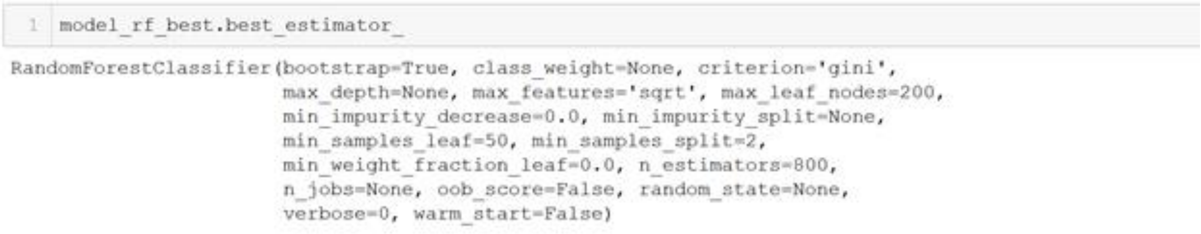
Running the Random Forest Model

The hyperparameter values to be passed to the final Random Forest model were determined by running a grid search with 5-fold cross validation with the below set of hyperparameters.

```
params_rf = {
                'n_estimators': [200, 400,600,800],
                'max_leaf_nodes': [50, 75, 100,200],
                'max_features' : ['auto','sqrt',0.25],
                 'min_samples_leaf': [50, 100, 200, 300]
                 }
 rf = RandomForestClassifier()
 model_rf_best = bestmodel(model_rf, params_rf)
```

To limit the usage of time for running the resource intensive grid search function the grid search function was fit using the sample training feature set and sample training target data sets that were created previously.

The below were the values of best parameters returned by the GridSearchCV function



```
model_rf_best.best_estimator_
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='sqrt', max_leaf_nodes=200,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=50, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=800,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Training the model on the complete data set using the chosen hyperparameter values

Once we got the best parameter values from grid search, the random forest model was again trained using the complete training data set (246008, 364).

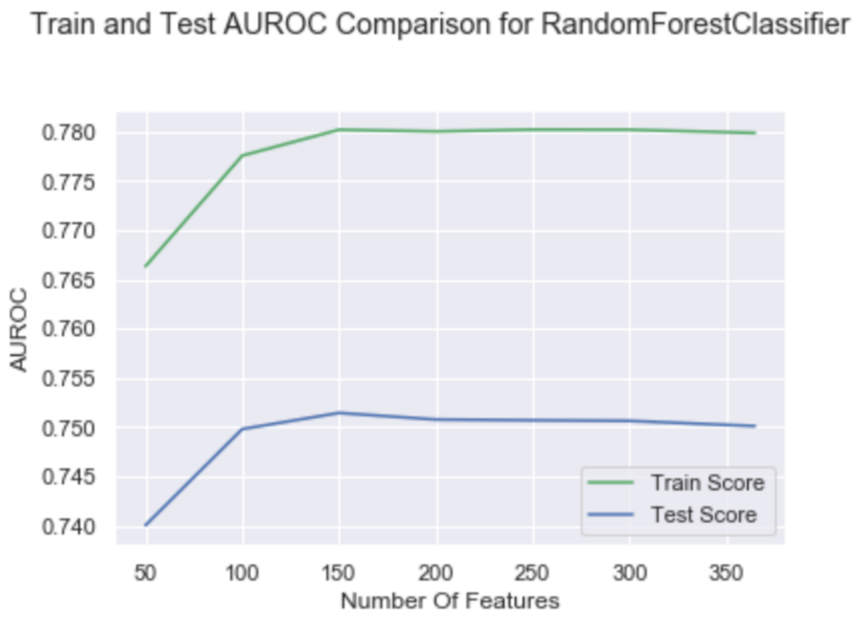
.

- **Training the model on training data**

  The Random Forest model with the best parameters were retrained on the training data created above for different number of top ranked features w.r.t feature importance, created during feature selection (Please refer to the **Feature Selection** section)

modeldetails(model_rf_best)

- **Evaluating the performance of the trained model with respect to train and test data and also with respect to # of top features**



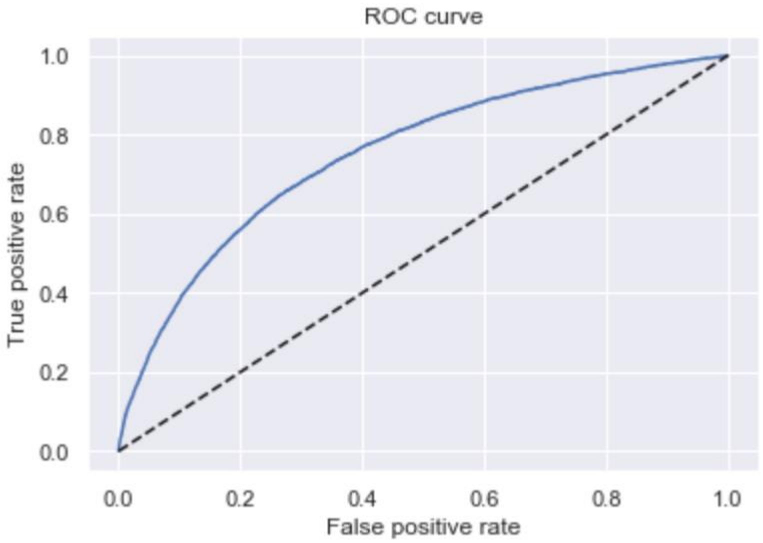
We can see that after 150 features, increasing # of features doesn't result in substantial increase in AUROC score, hence top 150 features appears to be optimum # of features to be passed to the final Random Forest model without compromising much on the model result.

rf_finalmodel_matrix =

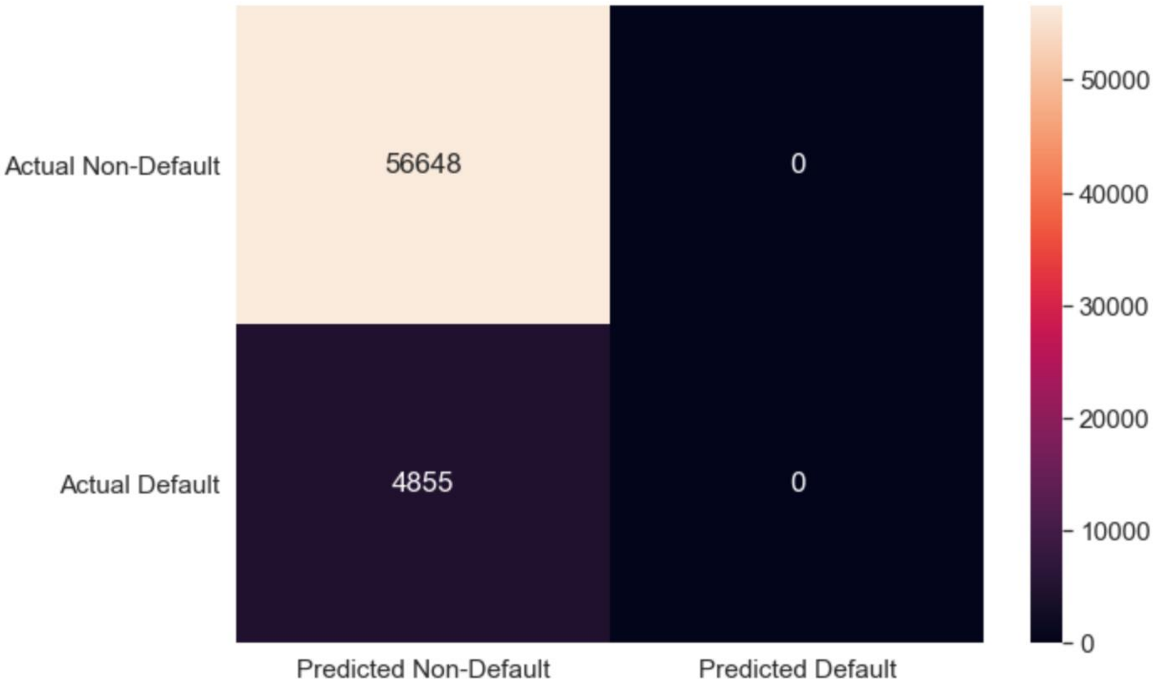
finalmodelperformance(150, model_rf_best.best_estimator_,0.5)

When the model was run on the unseen data it returned an AUC score of

AUC Score (Random Forest Model with 150 features): 0.7517



`Confusion Matrix (Random Forest Model with all 150 features)`



	Predicted Non-Default	Predicted Default
Actual Non-Default	56648	0
Actual Default	4855	0

50000
40000
30000
20000
10000
0

The problem with both the random forest models (one with all the features and the other one with 180 features) was that it was not at all able to pick any actual **default cases** in the test data set as can be seen from the respective confusion matrix. But the AUC score which considers both the false positives and true positives is reasonably good in this case because whereas the model couldn't pick up any of the default cases, it didn't misclassify any non-default cases either. In cases where the cost of rejecting an actual non-default is relatively a lot higher than giving loan to a customer who is going to default in future, this model will perform very good, but as in this case the business objective is **ensuring that deserving home credit clients are not rejected and at the same time ensuring minimum loss to home credit because of giving loans to future defaulters,** we have to balance out both the cases to achieve this objective. Hence, this model would not be a good choice for the case in hand.

Moreover, even if we only go by our chosen evaluation metric (AUC score) the logistic regression (`0.7560)` still performs better than the Random Forest model (`0.7517)`

Running the Light GBM Model

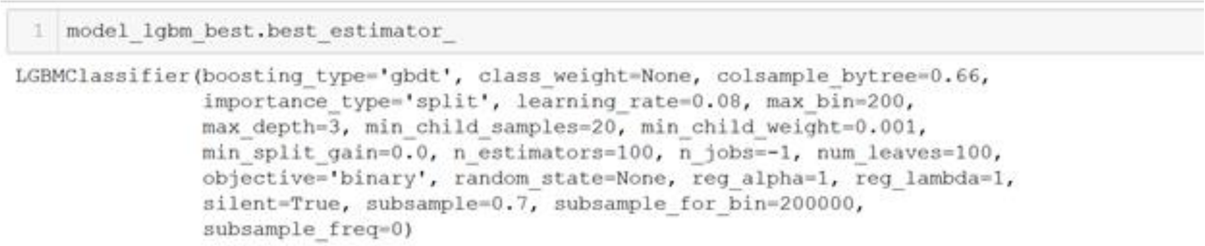
The hyperparameter values to be passed to the final Light GBM model were determined by running a grid search with 5-fold cross validation with the below set of hyperparameters.

```
params_lgbm = {

                    'num_leaves':[100,150,200,250],
                    'max_depth':[3,4,5,7],
                    'learning_rate':[.03,.05,0.08,1],
                    'max_bin':[100,200,300,400],
                    'boosting_type' : ['gbdt'],
                    'objective' : ['binary'],
                    'colsample_bytree' : [0.65, 0.66],
                    'subsample' : [0.7,0.75],
                    'reg_alpha' : [1,1.2],
                    'reg_lambda' : [1,1.2,1.4]
                 }
```

```
model_lgbm = lgb.LGBMClassifier()

model_lgbm_best = bestmodel(model_lgbm, params_lgbm)
```

To limit the usage of time for running the resource intensive grid search function the grid search function was fit using the sample training feature set and sample training target data sets.

The below were the values of best parameters returned by the GridSearchCV function



```
model_lgbm_best.best_estimator_
```

```
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=0.66,
               importance_type='split', learning_rate=0.08, max_bin=200,
               max_depth=3, min_child_samples=20, min_child_weight=0.001,
               min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_leaves=100,
               objective='binary', random_state=None, reg_alpha=1, reg_lambda=1,
               silent=True, subsample=0.7, subsample_for_bin=200000,
               subsample_freq=0)
```

Training the model on the complete data set using the chosen hyperparameter values

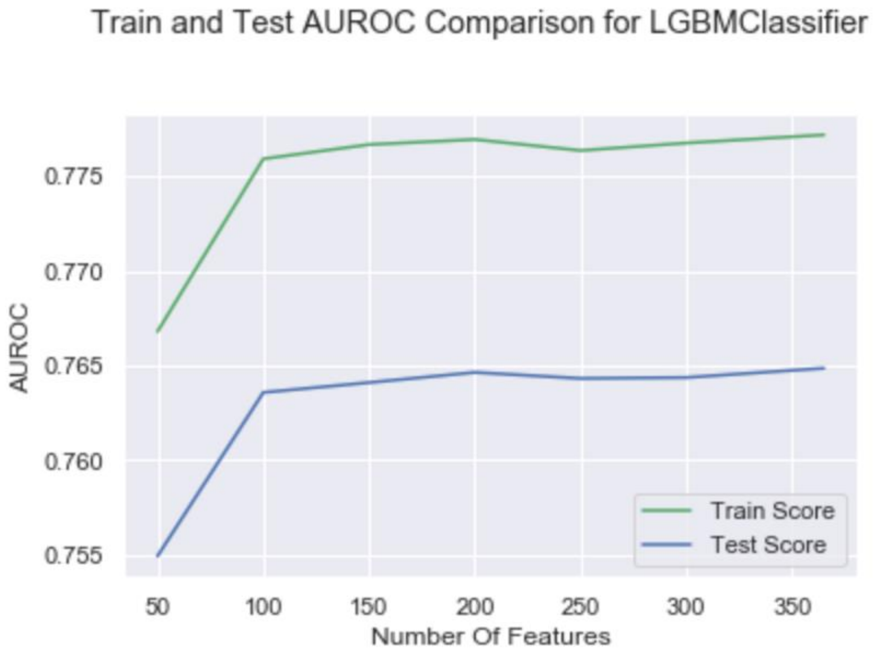
Once we got the best parameter values from grid search, the Light GBM model was again trained using the complete training data set (246008, 364).

- **Training the model on training data**

  The Light GBM model with the best parameters were retrained on the training data created above for different number of top ranked features w.r.t feature importance created during feature selection (Please refer to the **Feature Selection** section)

  modeldetails(model_lgbm_best)

- **Evaluating the performance of the trained model with respect to train and test data and also with respect to # of top features**



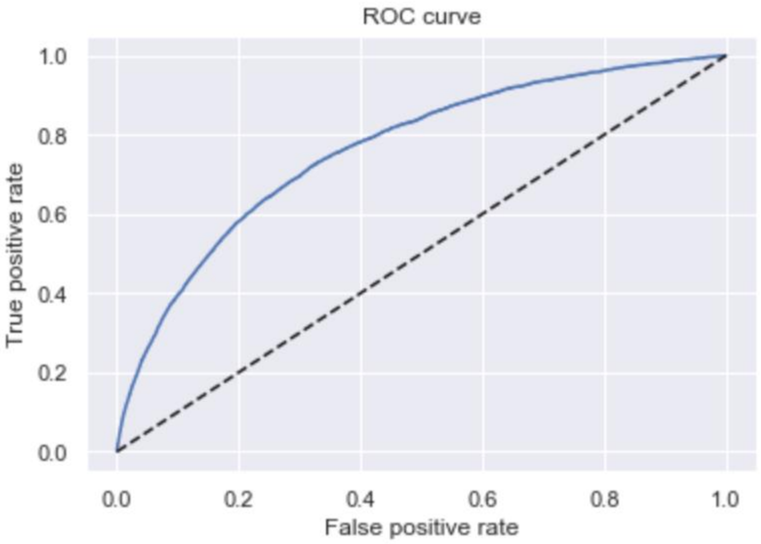
We can see that after 150 features, increasing # of features doesn’t result in substantial increase in AUROC score, hence top 150 features appear to be optimum # of features to be passed to the final Light GBM model without compromising much on the model result.

lgbm_finalmodel_matrix =

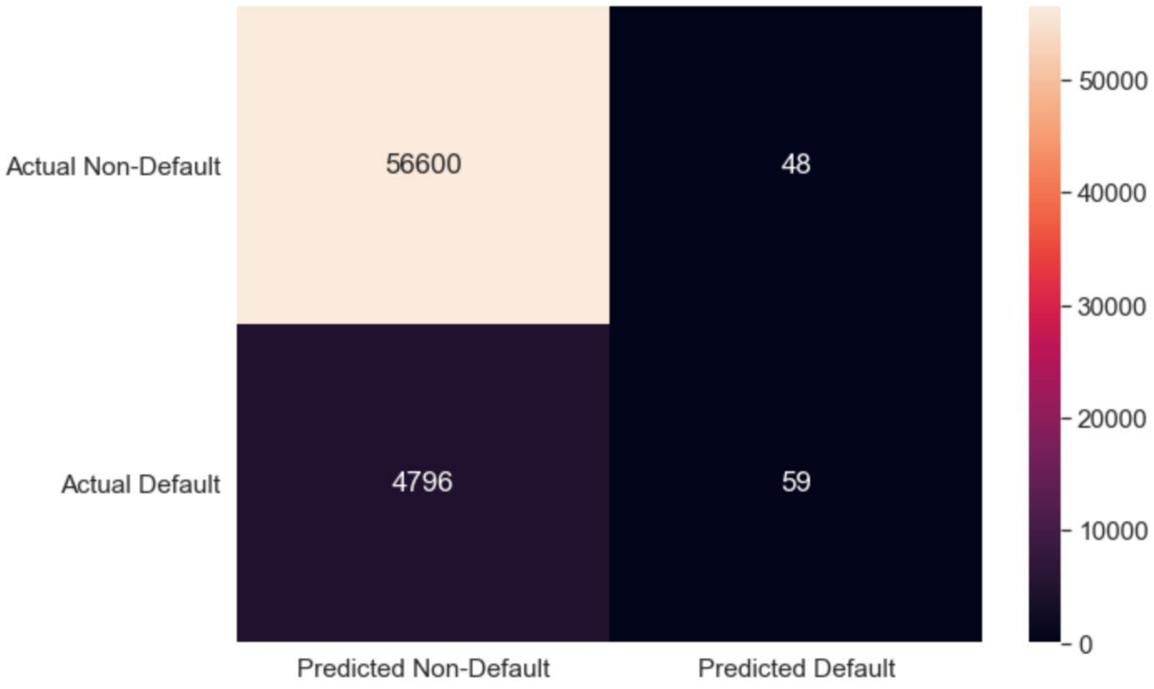
finalmodelperformance(150, model_lgbm_best.best_estimator_,0.5)

When the model was run on the unseen data it returned an AUC score of

AUC Score (Random Forest Model with 150 features): 0.7641



Confusion Matrix (Light GBM Model with all 150 features)



I chose the Light GBM model with 150 features along with the chosen hyperparameters as one of the contenders for the final model selection.

Running the XGBoost Model

The hyperparameter values to be passed to the final XGB model were determined by running a grid search with 5-fold cross validation with the below set of hyperparameters.

```
 params_xgb = {
                    'gamma': [0.5, 0.7, 0.9, 1],
                    'n_estimators': [800,1200,1400, 1600],
                     'objective': ['binary:logistic'],
                     'learning_rate': [0.03,0.05,0.06,0.08],
                     'max_depth': [2,4,6,8],
                     'seed' : [27],
                     'eval_metric': ['auc']
                      }
model_xgb = xgb.XGBClassifier()
model_xgb_best = bestmodel(model_xgb,params_xgb)
```

To limit the usage of time for running the resource intensive grid search function the grid search function was fit using the sample training feature set and sample training target data sets that were created previously.

The below were the values of best parameters returned by the GridSearchCV function.

```
{'eval_metric': 'auc', 'gamma': 0.5, 'learning_rate': 0.03, 'max_depth': 2,
'n_estimators': 800, 'objective': 'binary:logistic', 'seed': 27}
```

Training the model on the complete data set using the chosen hyperparameter values

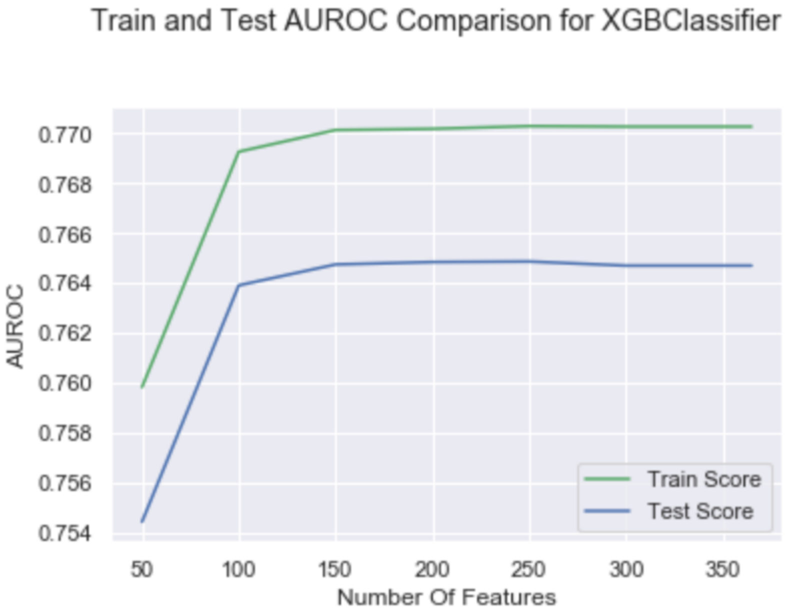
Once we got the best parameter values from grid search, the XGBoost model was again trained using the complete training data set (246008, 364).

- **Training the model on training data**

The XGBoost model with the best parameters were retrained on the training data created above for different number of top ranked features w.r.t feature importance created during feature selection (Please refer to the **Feature Selection** section)

modeldetails(model_xgb_best)

- **Evaluating the performance of the trained model with respect to train and test data and also with respect to # of top features**



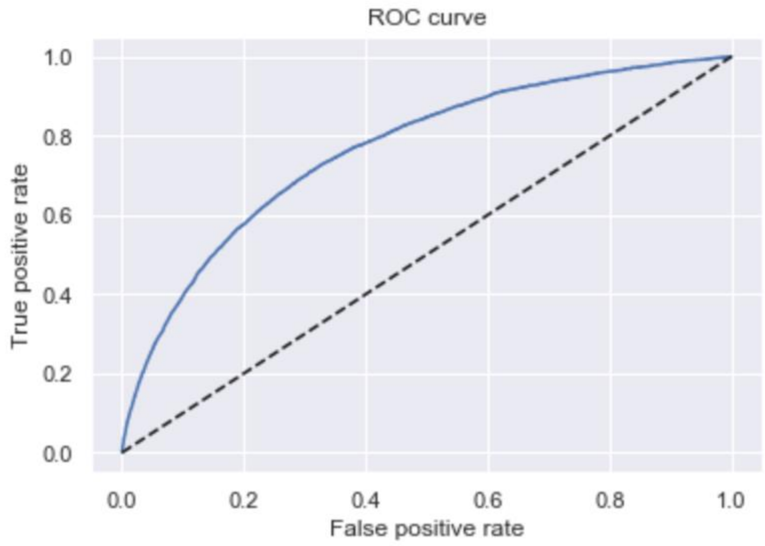
Similar to the case with most of the other models, we can see in this case as well that after 150 features, increasing # of features doesn't result in substantial increase in AUROC score, hence top 150 features appears to be optimum # of features to be passed to the final XGBoost model without compromising much on the model result.

xgb_finalmodel_matrix =

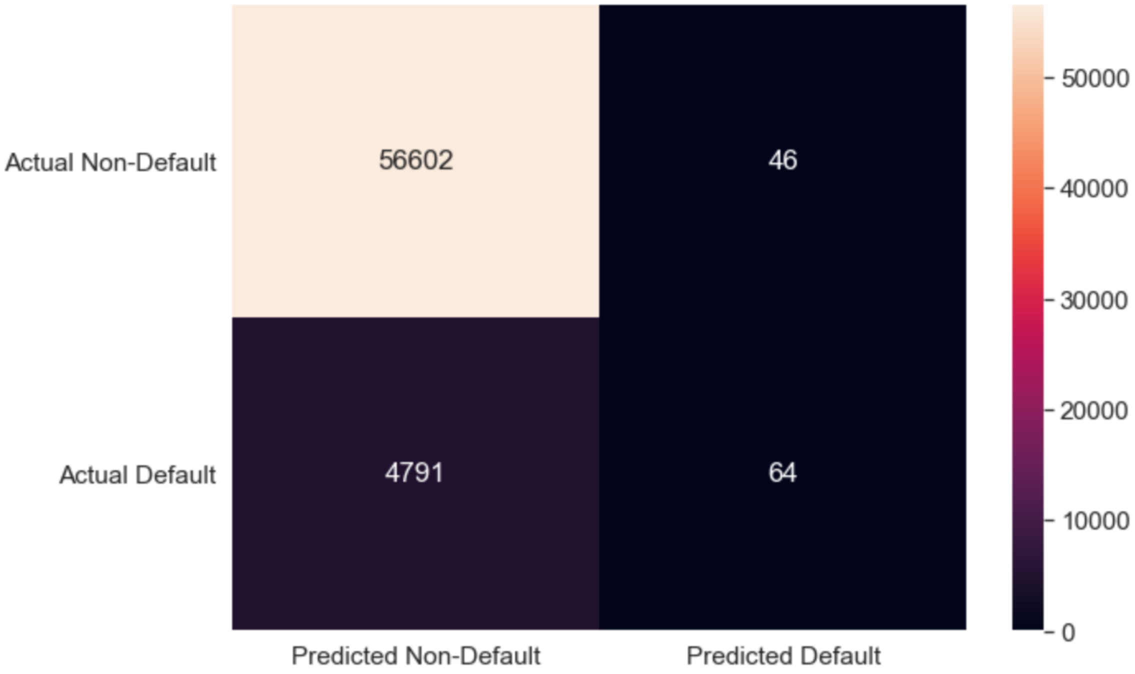
finalmodelperformance(150, model_xgb_best.best_estimator_,0.5)

When the model was run on the unseen data it returned an AUC score of

AUC Score (Random Forest Model with 150 features): 0.7647



Confusion Matrix (XGBoost Model with all 150 features)



Summary of all the models

After Running all the models, the below is the summary of how each model performed with respect to our evaluation metric (AUC score) on the test data set.

Model Name	AUC Score
Logistic Regression	0.7560
Random Forest	0.7517
Light GBM	0.7641
XGBoost	0.7647

Conclusion:

As we could see from the above table that the XGBoost model performed the best with respect to the AUC score. We also discussed, the model run time and model interpretability was not a big factor in choosing the model. Hence, I chose the XGBoost model in its current form as the final model to be used to predict the defaults for home credit.