Building a Model Which Predicts a Probability That a Given Customer Will Default on a Loan

Done By

ARNAOUT, BASSAM; Student ID: 002-25-3288

Presented To

Dr. Russell Butler

Dept. of Computer Science
Bishop's University
Sherbrooke, Canada

General Remark

The code attached “dataminingfinalproject.py” has been implemented in such a way that it will automate itself to selecting the best ML Model and do hyper-parameter tuning to reach the optimal value for ROC-AUC.

Summary

Which algorithm did we use and why?

This is a classification-type problem, for this we have used the following 9 models as a start:

1. Logistic Regression
2. Linear Discriminant Analysis
3. KNN
4.Gaussian Naive Bays
5. Decision Tree
6. Support Vector Machine
7. Multi-layer Perceptron classifier.
8. Gradient Boosting
9. Extra Trees Classifier

At early stage & after adjusting the Data, we have used "Spot-Check Classification Algorithms". This algorithm uses 10-fold cross validation.

The result came up that "**Gradient Boosting**" was the **best model** in terms of **highest roc_auc** value and we based our building the model on it.

How certain are we of the results?

After we have fed the selected Model "Gradient Boosting" with the data, trained it and fine tune it's hyper-parameters, we had reached around above 72% of Accuracy rate & on average of 68% of roc_auc (on Train, Test & Validated Data)

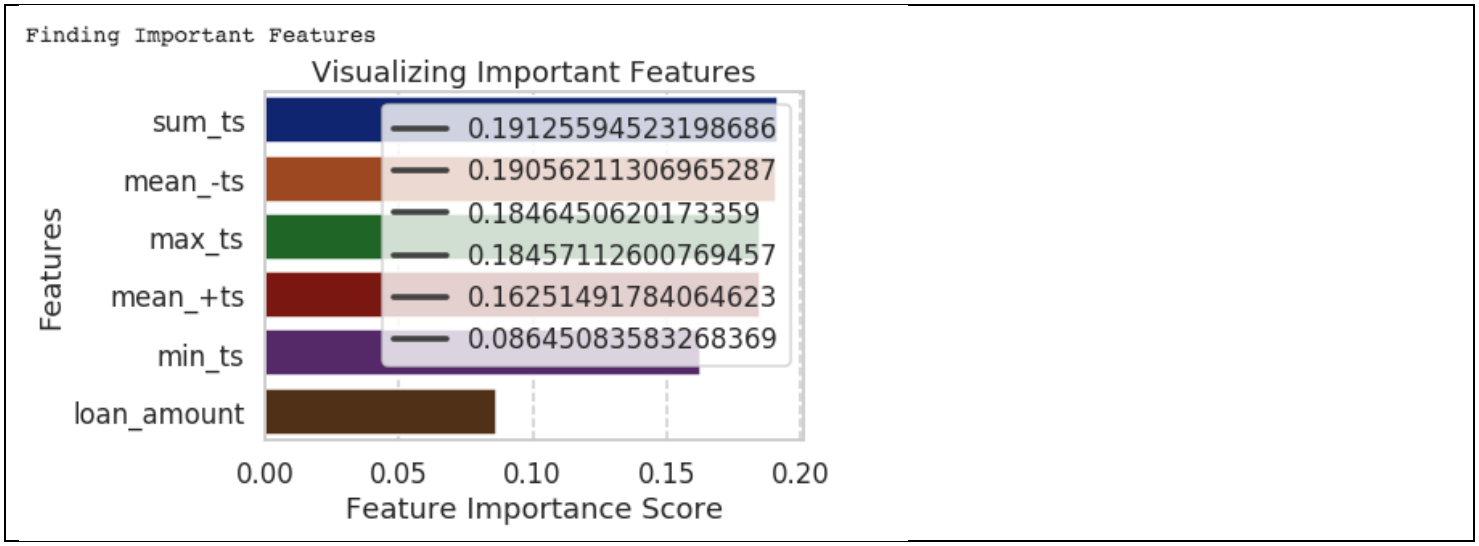
Is there a subset of potential customers that are very safe to lend to? A subset that is very dangerous?

We can say those customers who have their probabilities < 30% (isDefault=1 Probability), will be able to take loans and pay them on a timely manner. 70% of time, they will be able to pay back the loan.

Customers who have their probabilities more than 30% will not be able to take loans because there is 70% risk of not paying back the loan.

What features of the dataset best predict isDefault?

The most important features that contributed in Building the Model were:
* sum_ts : sum of all transactions per customer
* mean_-ts: mean(average) value of all negative transactions per customer
* mean_+ts: mean(average) value of all positive transactions per customer
* max_ts: maximum value among all transactions happened per customer
* min_ts: minimum value among all transactions happened per customer
* loan_amount : amount value requested for a loan

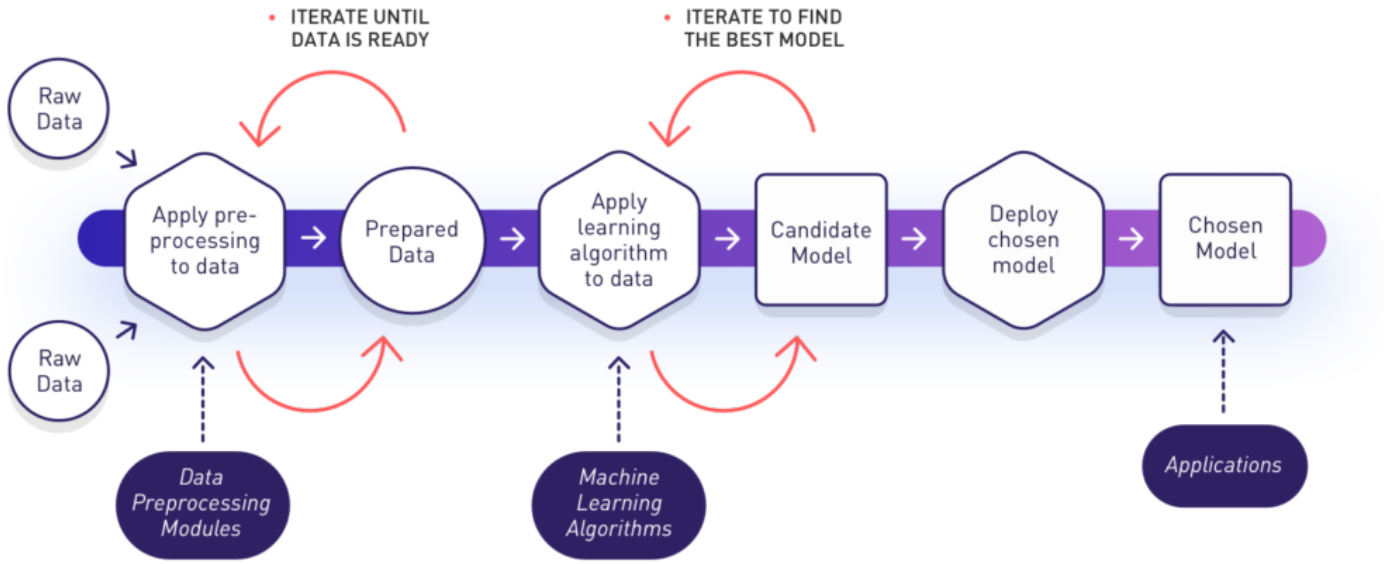


As we can see by running "Finding Important Features Using RandomForestClassifier", we can tell that all the above features are equally important except "loan_amount" attribute and hence can be neglected in Building the Model.

Do these features make sense intuitively? Justify our use of the features

Yes, these are equally important for the model training.
Intuitively speaking, all are very important (except loan_amount feature) because each one describe the customer's activity at bank in terms of transactions like how much maximum certain customer had deposited, had withdrawn. What is his current balance now? And what is his average of withdrawal and/or deposits. All these features have contributed a lot in building the model.

General Steps to Build Best Model for a Data



The basic data looks like the below:

Bank Customer Transactions Dataset:

	id	dates	transaction_amount	days_before_request	loan_amount	loan_date	isDefault
0	[id, 00076211-9BBD-4E06-82C2-E9FB7B102964-750]	[dates, [2017-05-19, 2017-05-29, 2017-05-31, 2...	[transaction_amount, [[-60.0], [0.02], [0.01],...	[days_before_request, [[79.0], [79.0], [76.0],...	[loan_amount, 750]	[loan_date, 2017-08-16]	[isDefault, 1]
1	[id, 0009107F-8F57-441C-A590-5773992261A9-500]	[dates, [2017-11-20, 2017-11-20, 2017-11-20, 2...	[transaction_amount, [[-68.93], [-23.7], [-0.5...	[days_before_request, [[12.0], [12.0], [12.0],...	[loan_amount, 500]	[loan_date, 2018-02-17]	[isDefault, 1]
2	[id, 000B47CB-514F-446C-B7BE-2F2D304E2F4D-1000]	[dates, [2017-05-29, 2017-05-29, 2017-05-29, 2	[transaction_amount, [[-23.5], [-9.78], [-14.2...	[days_before_request, [[87.0], [87.0], [87.0],...	[loan_amount, 1000]	[loan_date, 2017-08-25]	[isDefault, 0]

We have worked in the transactions per each customer and concluded the following attributes out of it.
This will be our final data to use for building our Model on it.

	mean_+ts	mean_-ts	sum_ts	max_ts	min_ts	loan_amount	isDefault
0	440.055172	-78.789942	-711.48	3082.73	-562.00	750.0	1.0
1	178.169900	-59.946844	911.98	1507.26	-1507.26	500.0	1.0
2	921.300000	-56.448514	366.63	1639.35	-811.23	1000.0	0.0
3	197.990000	-46.844850	332.54	1000.00	-900.00	250.0	0.0
4	375.659231	-22.771498	-285.56	806.39	-320.09	500.0	0.0
...	...	...	...	...	...	...	...

After analyzing the data and applying some filtering and data manipulation on it such as "Data Scaling" and "Selecting Most important Features", "Skewing Adjustment" **(all this shown in the code attached),** we found that none of the above techniques enhance the model performance later on.

So the above dataset is the final one we have adopted to feed it into the 9 classification-models, training them, doing best model selection according to best AUC value, tuning the selected Model and finally apply it to predict the probabilities of (isDefault = 1, i.e customer will default from paying his loan) for the last 5000 customers (who has missing isDefault value).

Create a Validation/Test Dataset

****

Create a Training, Validation & Test Dataset:

Training set has 6700 samples.
Validation set has 1650 samples.
Testing set has 1650 samples.

Spot-Check Classification Algorithms

We applied such technique on the 9 classification models and result was as below:

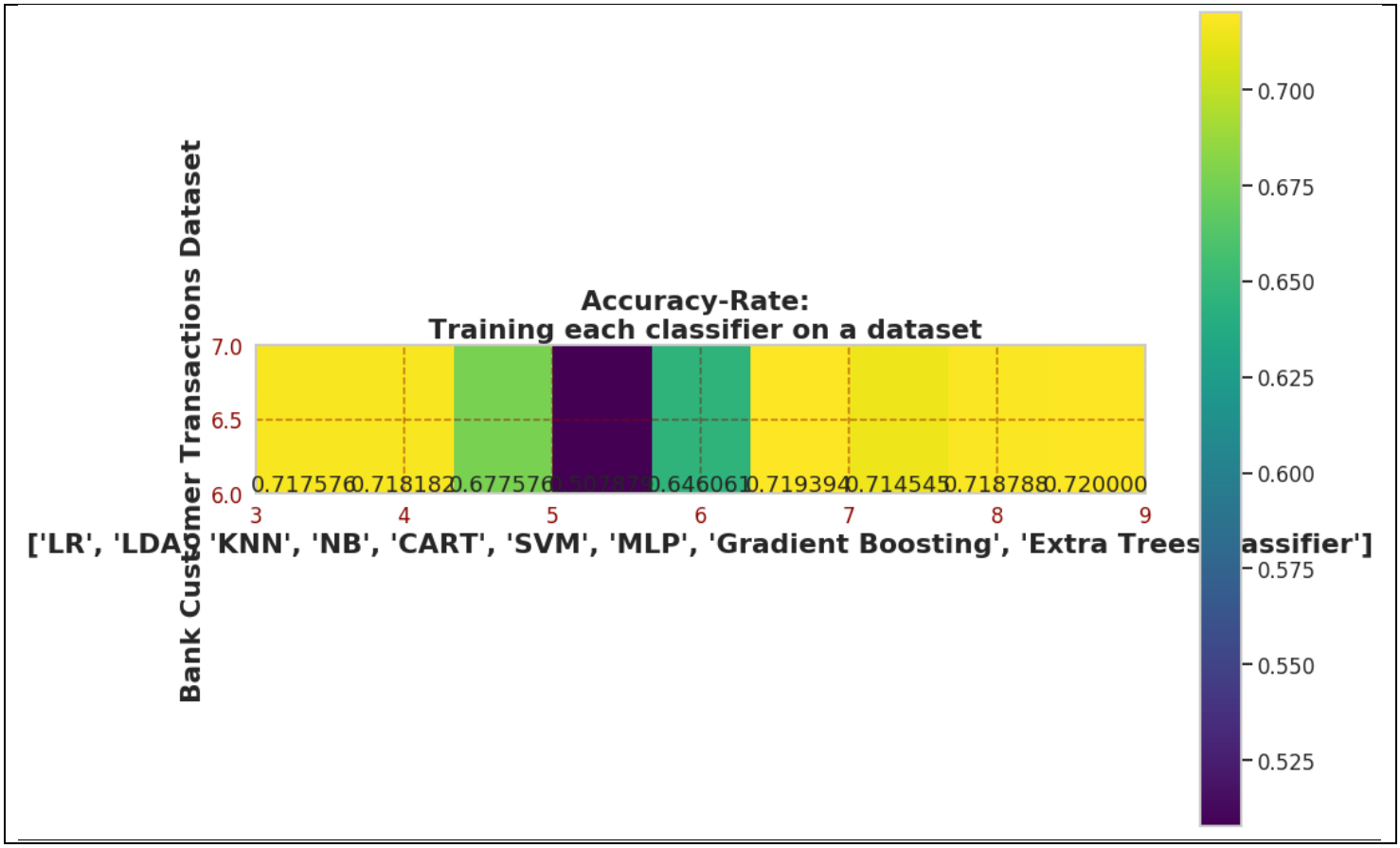
Applying Spot-Check Classication Algorithms:

LR| auc_mean:0.625303 auc_std(0.006690)
LDA| auc_mean:0.626329 auc_std(0.003766)
KNN| auc_mean:0.553562 auc_std(0.006611)
NB| auc_mean:0.623702 auc_std(0.002436)
CART| auc_mean:0.528306 auc_std(0.006222)
SVM| auc_mean:0.502650 auc_std(0.001275)
MLP| auc_mean:0.564056 auc_std(0.011143)
Gradient Boosting| auc_mean:0.651407 auc_std(0.003625)
Extra Trees Classifier| auc_mean:0.619242 auc_std(0.004632)

This script took 0.36 minutes to complete

We can see the Gradient Boosting gave the best roc_auc value, so we will use this model & will start with our hyper-parameters tuning on this model only.

Plotting Accuracy Matrix (Dataset vs. Classifiers)



Select Best Model

****

Selecting Best Model:

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='auto',
                           random_state=None, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

Tuning Gradient Boosting Classifier with Grid Search CV & Random Search

Step 1- Find the number of estimators for a high learning rate	Best AUC : 0.653573 using {'n_estimators': 100}
Step 2: Tune max_depth and min_samples_leaf	Best AUC : 0.660936 using {'max_depth': 2, 'min_samples_leaf': 4}
Step 3: Tune max_features	Best AUC : 0.661356 using {'max_features': 4}
Step 4: Tune subsample	Best AUC : 0.663111 using {'subsample': 0.96}
Step 5: Reducing Learning Rate (using Random Search)	Best accuracy is 0.6654857940131913

The best roc_auv accuracy reached after hyper-tuning the model is 0.6654857940131913

Final Tuned Model will be:

****

load the model from disk:

GradientBoostingClassifier(criterion='friedman_mse', init=None,
learning_rate=0.07630828937395717, loss='deviance',
max_depth=2, max_features=4, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=4, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100,
n_iter_no_change=None, presort='auto',
random_state=None, subsample=0.96, tol=0.0001,
validation_fraction=0.1, verbose=0,
warm_start=False)

Make Predictions: Precision, recall, F1score for the Model

Prediction on Validation data:

```
**********
Prediction on Validation data:

[[1184    0]
 [ 463    3]]
              precision    recall  f1-score   support

         0.0       0.72      1.00      0.84      1184
         1.0       1.00      0.01      0.01       466

    accuracy                           0.72      1650
   macro avg       0.86      0.50      0.42      1650
weighted avg       0.80      0.72      0.60      1650
```

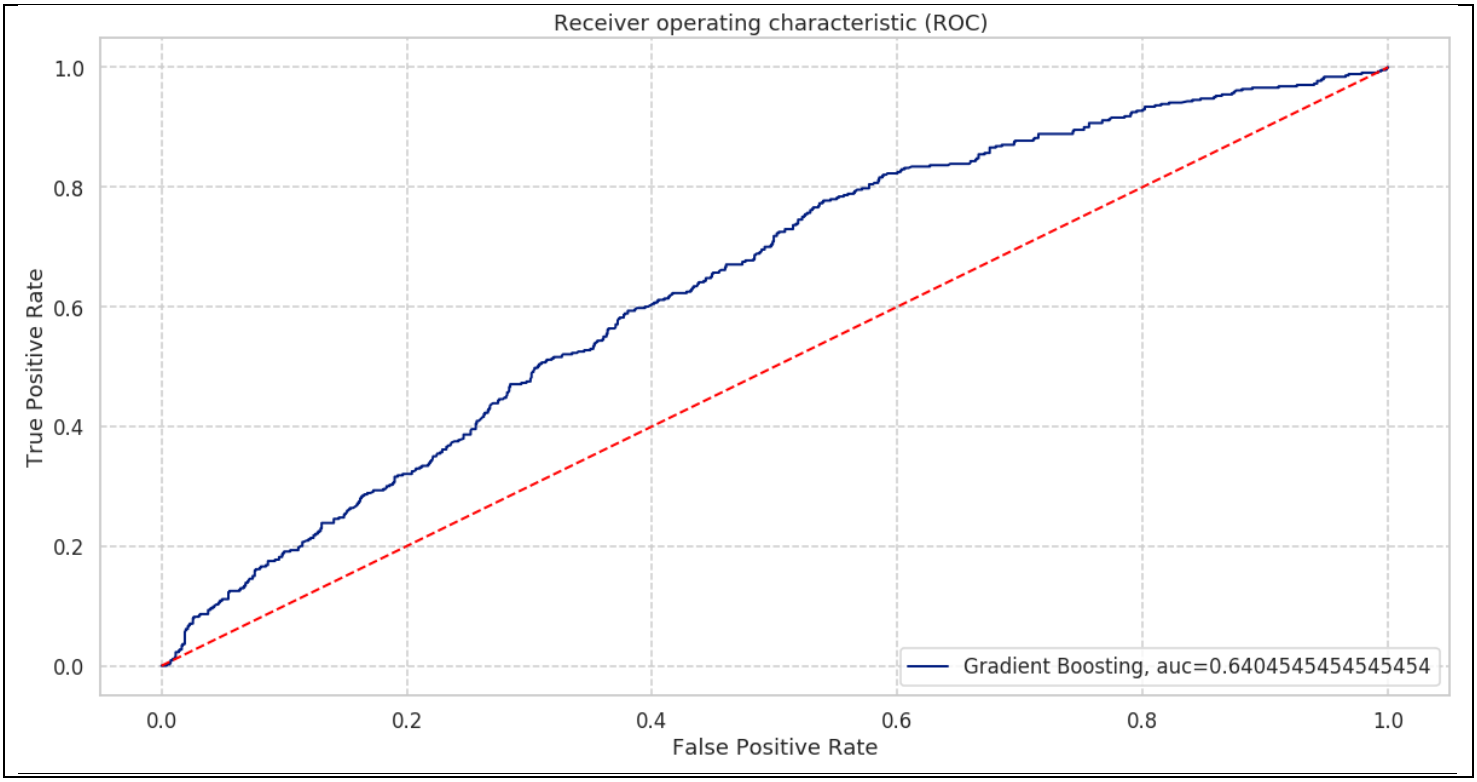
Prediction on Test data:

```
**********************************************************************************
Prediction on Test data:

[[1202    8]
 [ 438    2]]
              precision    recall  f1-score   support

         0.0       0.73      0.99      0.84      1210
         1.0       0.20      0.00      0.01       440

    accuracy                           0.73      1650
   macro avg       0.47      0.50      0.43      1650
weighted avg       0.59      0.73      0.62      1650
```



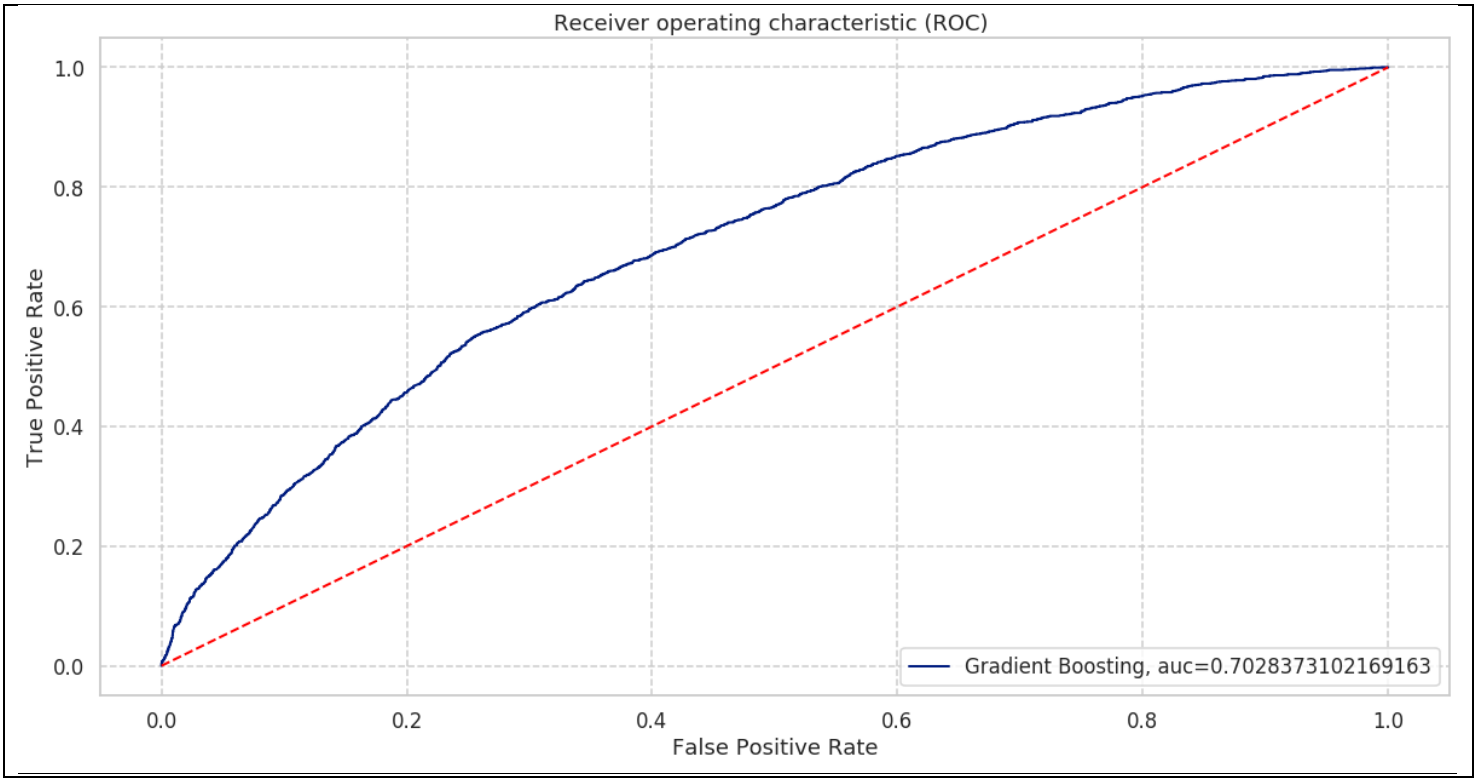
Prediction on Train data:

```
**********************************************************************
Prediction on Train data:

[[4895   11]
 [1773   21]]
              precision    recall  f1-score   support

         0.0       0.73      1.00      0.85      4906
         1.0       0.66      0.01      0.02      1794

    accuracy                           0.73      6700
   macro avg       0.70      0.50      0.43      6700
weighted avg       0.71      0.73      0.63      6700
```



Printing Probabilities

```
**********************************************************************************************
Printing first 10 records from last 5000 data for the Probabilities that the customer will default
from paying his loan (isDefault=1):

A9C3F4CA-66BF-44EE-8C38-1C50D0751CD1-500, Predicted=0.3792736702706167
A9C93D6B-8625-48EE-9E05-EB63256DEEA6-500, Predicted=0.261546297052941
A9D12F62-35D5-44DE-8446-B8C2E4A5E938-750, Predicted=0.1749354790964226
A9D3EC9F-B3EB-4A56-8BC6-4614A1C42FE2-600, Predicted=0.19630006038867293
A9D54EF9-3CBA-4481-80B2-59FD92BB60D9-700, Predicted=0.3032890171662117
A9D8D5D1-C59F-4C78-AF37-1DA4A9D4D1FC-750, Predicted=0.23193142225287908
A9D9A46A-D825-4F84-9E21-8F09352E93AF-600, Predicted=0.4057569150585736
A9DDD331-06FB-41D9-ADDD-318E2530ECD8-750, Predicted=0.222544429174621
A9E034FA-C91A-45F9-B69E-F765A67E3C2E-750, Predicted=0.17805967878192547
A9E36B32-D292-4C8A-948B-B33CB1EDA176-500, Predicted=0.34979975610642855

Please check "Bassam_Arnaout.txt" for all probability output for last 5000 Customers.


**********************************************************************************************
 END
**********************************************************************************************
```