Out[1]:

The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

Supervised Learning Capstone

A Nation of Debtors

Farooq Hassan

November 22, 2019

Purpose

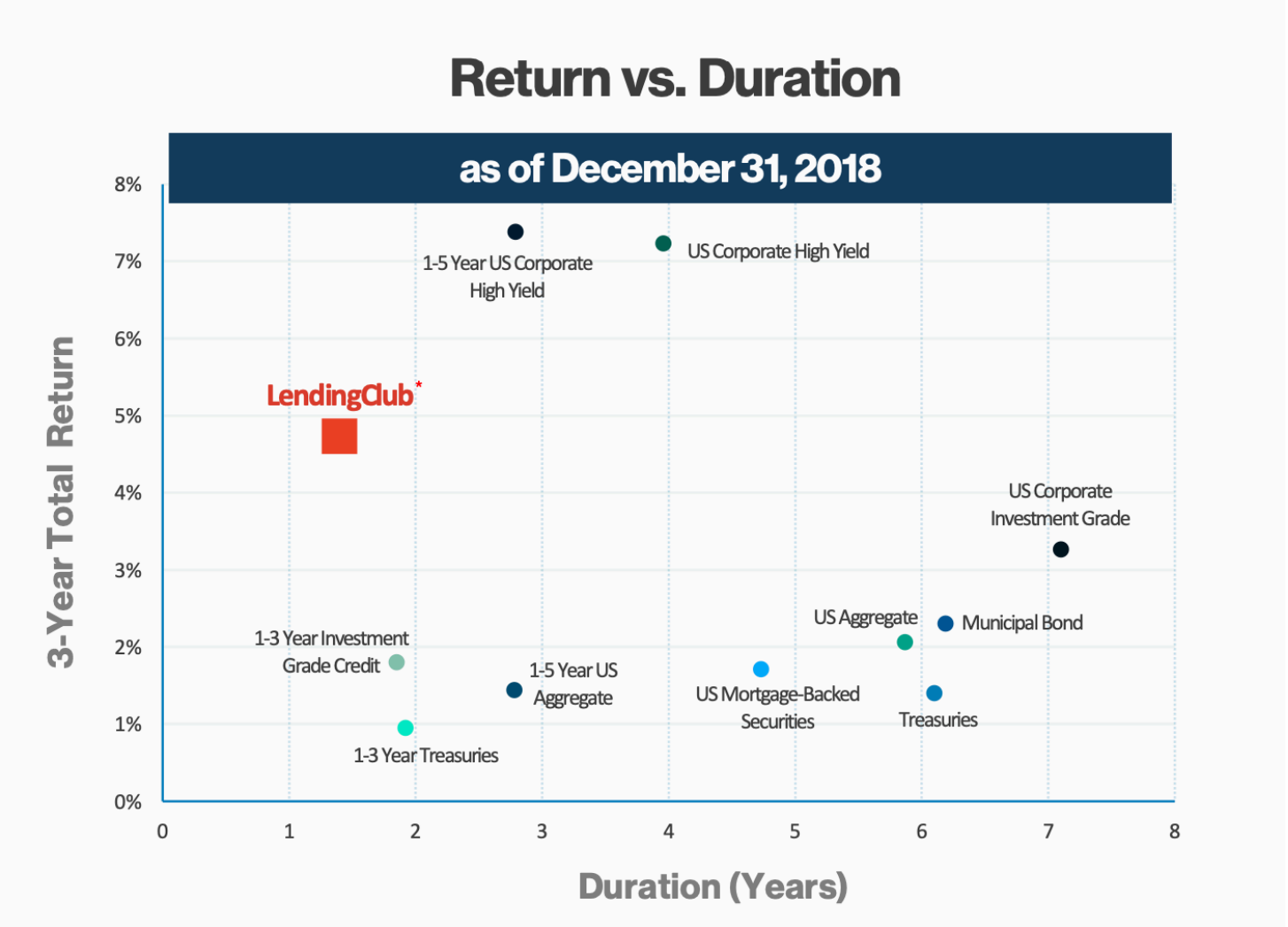
Goal

Improve the gross margin of LendingClub through the identification of problematic loans at the time of issuance for further analysis and/or allocation to securitization facilities.

Objective

Develop a supervised learning model that can effectively predict whether a loan will be fully paid off or charged off at the time of loan approval.

Introduction



Source : LendingClub, Brismo, Morningstar, and Barclays Live, as of 12/31/2018

LendingClub (NYSE: LC) is a peer-to-peer lending platform that pairs insititutional and individual investors with borrowers.

LendingClub's top 10 institutional investors hold 54% of the loans outstanding.

Overview

The business model is dependent upon credit creation in the US as well as an adequate Net Interest Margin (NIM).

LendingClub generates the majority of their revenues from loan origination fees and loan servicing fees. Borrowers pay an origination fee of 0% to 6% of the total loan amount as issuance. The company charges a 1% servicing fee on the monthly loan payment. If a borrower defaults on their loan, LendingClub does not provide any reimbursement. LendingClub holds approximately 40% of the loans orignated on the platform as investments. Therefore improving defaults rates is beneficial to the company as well investors.

Notes: 1.0TN revolving, 1.6TN is student loan, and 1.2TN is motor vehicle Record loan originations of 3.3 billion, up 16% year-over-year. Record Net Revenue of 204.9 million, up 11% year-over-year.

Data driven innovation in demand generation helped grow applications, improve conversion and retention, and drive business model efficiency. 71% of customers went from application to approval within 24 hours, up from 60% in the third quarter of 2018, helping increase LendingClub's Net Promoter Score to 80.

Consumer credit increased at a seasonally adjusted annual rate of 5 percent during the third quarter.

Revolving credit increased at an annual rate of 2-1/4 percent, while nonrevolving credit increased at an annual rate of 6 percent.

In September, consumer credit increased at an annual rate of 2-3/4 percent.

Under a loan sale agreement, WebBank may sell us loans without recourse two business days after WebBank originates the loan.

The platform uses proprietary risk algorithms that leverage behavioral data, transactional data and employment information to supplement traditional risk assessment tools, such as FICO scores, to assess a borrower’s risk profile. For certain loans, our verification processes and analysts then verify the borrower’s identity, income or employment by connecting with various data providers to determine whether to approve the loan request, in accordance with the issuing banks’ credit policy *italicized text*

Notes: $10.8bn in issuance, 75% of revenues is generated from origination fees and 15% from servicing fees. The spread on origination is 4.8% - 2.8% or 2.0% while servicing fees ..... The company's Top 10 customers account for 58% of the revenues The company holds approximately 40% of loans for investment at a NIM (Net Interest Margin) of 0.8%.

Duration = Present value of a bond's cash flows, weighted by length of time to receipt and divided by the bond's current market value.

Loans range from USD1,000 to USD40,000 with an average duration of 16 months and default rate of 6.5% on 36 month loans.

The company is the market leader in personal loans – a 140 billion+ industry and the fastest growing segment of consumer credit in the United States – and has an estimated potential immediate addressable market opportunity of more than 445 billion.

Data

The data is compiled of 15 csv files downloaded from LendingClub's website by creating an account. The data can be accessed at the following URL: https://www.lendingclub.com/info/download-data.action (https://www.lendingclub.com/info/download-data.action)

The data is from Q1 2016 to Q3 2019 and is comprised of 1.7MM rows with 150 columns.

The underlying data set used to train and test the models was a list of customers who had applied and been approved for a loan from LendingClub between January 2016 and September 2019.

A data dictionary library was provided in excel format that provided a brief description of each variable.

Read CSV & Parquet files

Choose Files no files selected

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving pc3.parquet.gzip to pc3.parquet (1).gzip
```

CSV

```
rows:  1763107
columns:  150

column names: id                          object
member_id                float64
loan_amnt                float64
funded_amnt              float64
funded_amnt_inv          float64
                          ...
settlement_status         object
settlement_date           object
settlement_amount        float64
settlement_percentage    float64
settlement_term          float64
Length: 150, dtype: object


<class 'pandas.core.frame.DataFrame'>
Int64Index: 1763107 entries, 0 to 143036
Columns: 150 entries, id to settlement_term
dtypes: float64(111), object(39)
memory usage: 2.0+ GB
```

Out[0]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	t
0	75243591	NaN	30000.0	30000.0	30000.0	mo
1	75284285	NaN	25000.0	25000.0	25000.0	mo
2	75358344	NaN	6950.0	6950.0	6950.0	mo
3	75369482	NaN	28000.0	28000.0	28000.0	mo
4	75644316	NaN	7000.0	7000.0	7000.0	mo

Exploratory Data Analysis

The data was reviewed by first examining the Dataset Dicitionary Library provided by LendingClub and later reading the 2019 10-K (Annual) and most recent 10-Q (Quarterly) filings.

Description of Features:

There were a total of 139 continuous variables and 11 categorical variables.

Data Cleaning:

In order to make an objective comparions of the loans, the data for individual loans were reviewed while joint loans and their attributable features were removed.

A subjective best attempt was conducted to remove all columns that contained values after the loan was approved.

Features that were not necessary such as specific ids, post loan information, redundant and investor only were also dropped.

This reduced the potential features from 149 to 61.

EDA Continued

Data Cleaning and Transformating:

Cleaning up the data and transforming them to get them ready for analysis.

- Changed "Fully Paid" to 0 making it numeric
- Changed "Charged Off" to 1 making it numeric
- NaNs were then replaced with zeros
- Removed special characters from columns avoiding errors created by special characters altogether
- Converted string objects to datetime
- Converted column values to floats or integers

Feature Engineering

- The categorical variables were one hot encoded
- A new feature was added to measure the change in the FICO score

- Total amount of all loans: 10.65bn
- Average loan amount: USD 14,463
- Median loan amount: USD 12,000

EDA

Out[0]:

```
Current               959549
Fully Paid            583498
Charged Off           152926
Issued                 27913
Late (31-120 days)     23217
In Grace Period         8094
Late (16-30 days)       6208
Default                 1672
Name: loan_status, dtype: int64
```

Out[0]:

```
(736424, 150)
```

Out[0]:

```
 36 months    575171
 60 months    161253
Name: term, dtype: int64
```

```
next_pymnt_d                                   736424
member_id                                      736424
desc                                           736405
orig_projected_additional_accrued_interest     732786
payment_plan_start_date                        730893
                                                ...
delinq_amnt                                         0
mo_sin_old_rev_tl_op                                0
mo_sin_rcnt_rev_tl_op                               0
mo_sin_rcnt_tl                                      0
id                                                  0
Length: 150, dtype: int64
```

Out[0]:

```
(736424, 147)
```

Out[0]:

	index	id	member_id	loan_amnt	funded_amnt	funded_amnt_i
0	0	75243591	NaN	30000.0	30000.0	3000(
1	1	75284285	NaN	25000.0	25000.0	2500(
2	2	75358344	NaN	6950.0	6950.0	695(
3	3	75369482	NaN	28000.0	28000.0	2800(
4	4	75644316	NaN	7000.0	7000.0	700(

```
Fully Paid     583498
Charged Off    152926
Name: loan_status, dtype: int64

Total amount of all loans: $10651156325.00
Average loan amount: $14463.35
Median loan amount: $12000.00
```

Distribution of Features

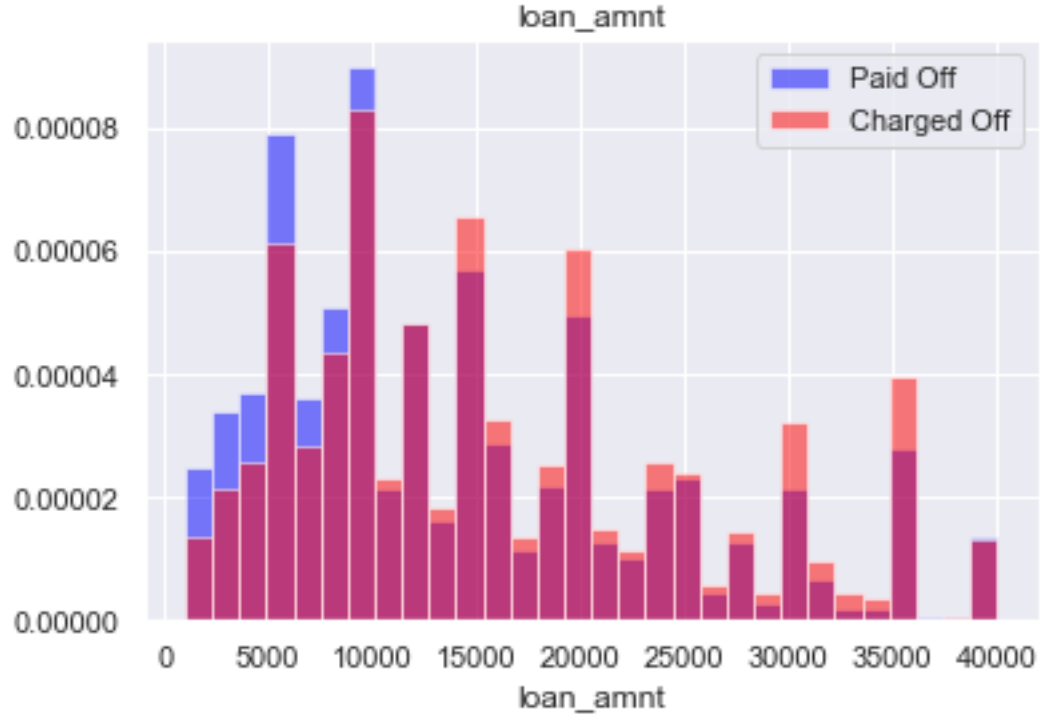
Out[0]:

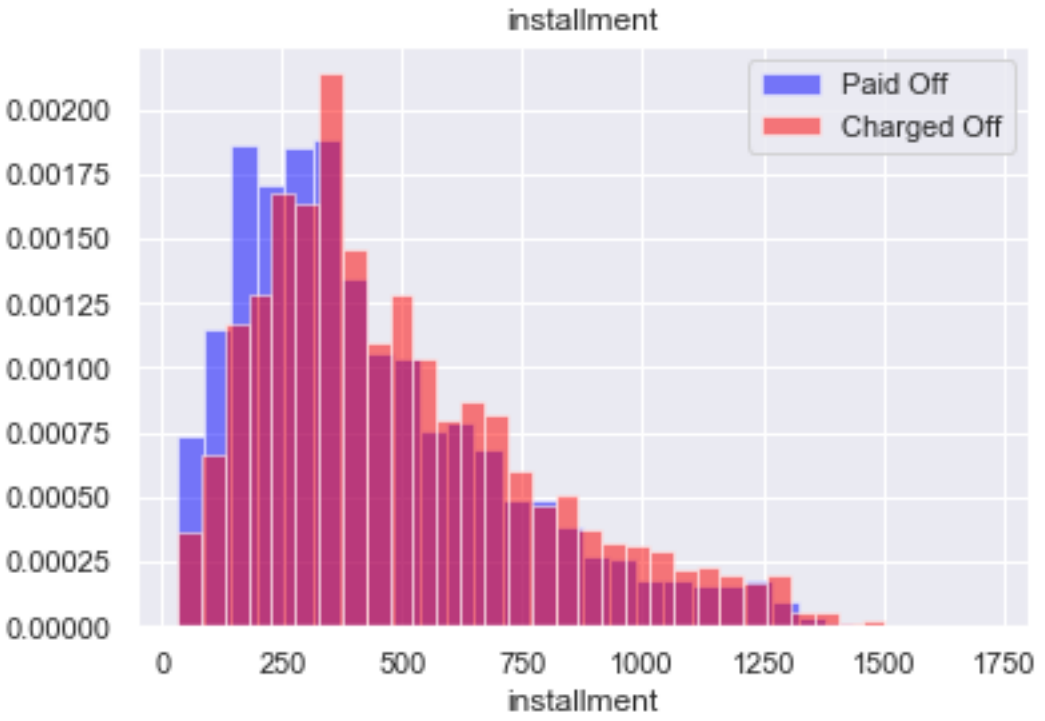
emp_length	default_rate	count
1 year	0.230234	2011
10+ years	0.184554	9970
2 years	0.193704	2700
3 years	0.219262	2440
4 years	0.208174	1835
5 years	0.204444	1800
6 years	0.176106	1289
7 years	0.198556	1108
8 years	0.210849	1143
9 years	0.204280	1028
< 1 year	0.214286	2562

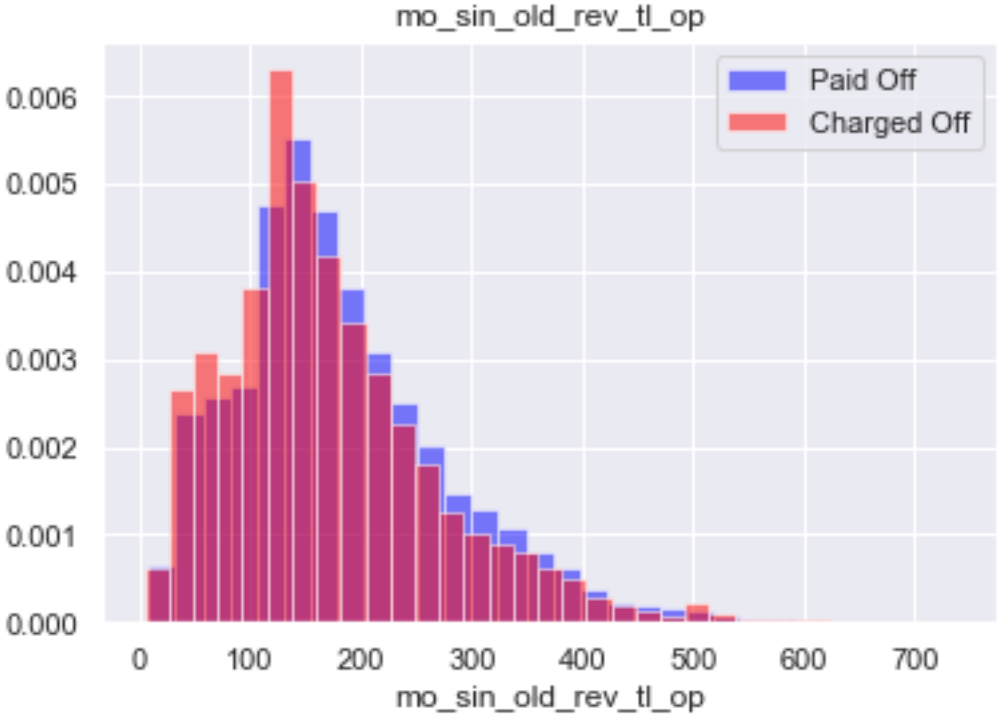
Out[0]:

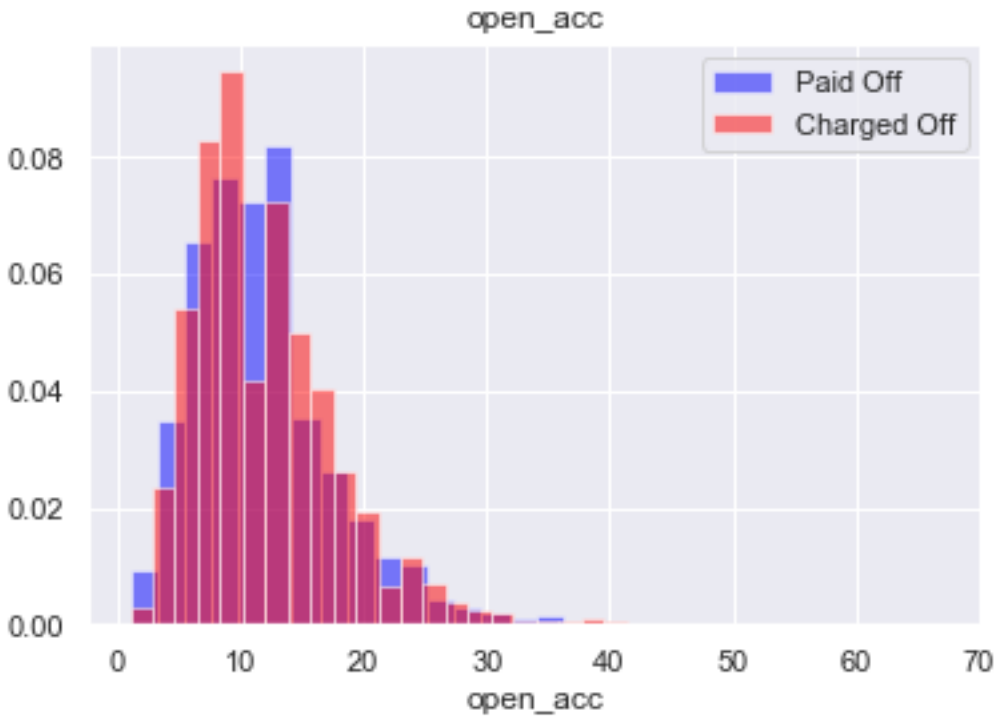
home_ownership	default_rate	count
ANY	0.210526	19
MORTGAGE	0.167733	14845
OWN	0.209769	3685
RENT	0.250808	11451

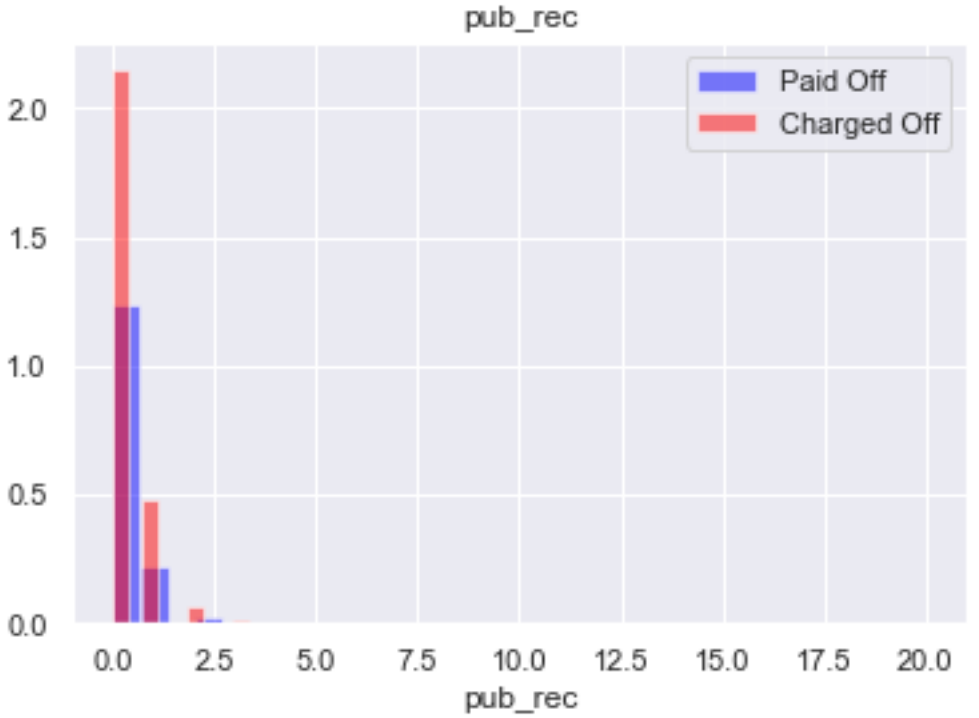

employment verification status
0.25
0.20
0.15
0.10
0.05
0.00
default rate
Not Verified
Source Verified
Verified
verification status

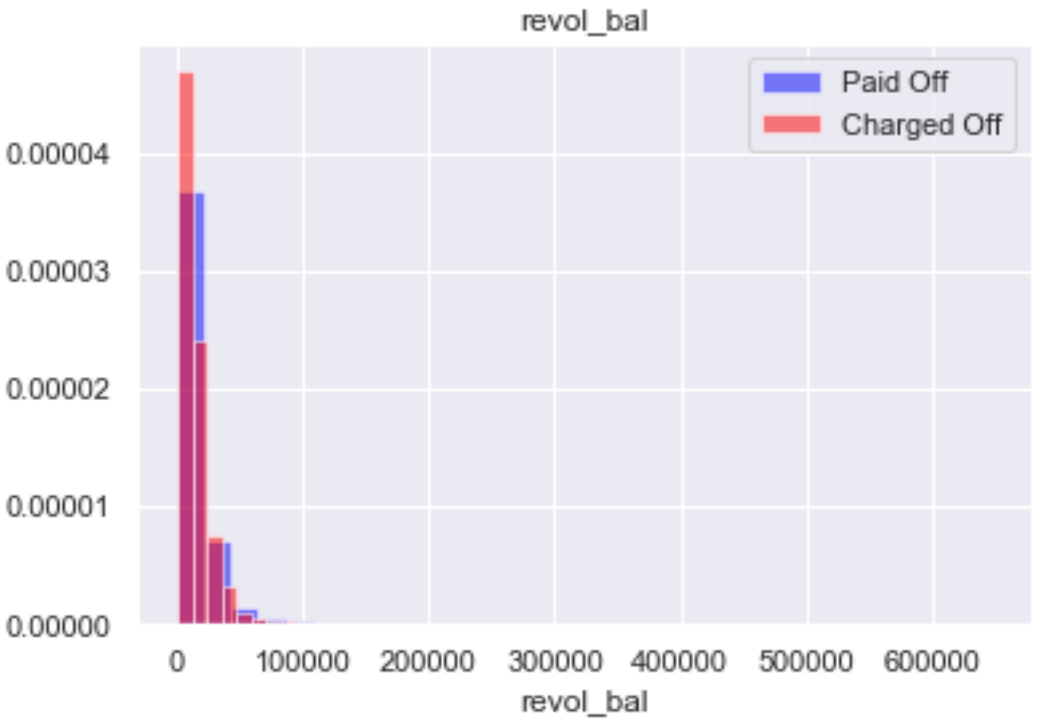

loan_amnt
Paid Off
Charged Off
0.00008
0.00006
0.00004
0.00002
0.00000
0
5000
10000
15000
20000
25000
30000
35000
40000
loan_amnt

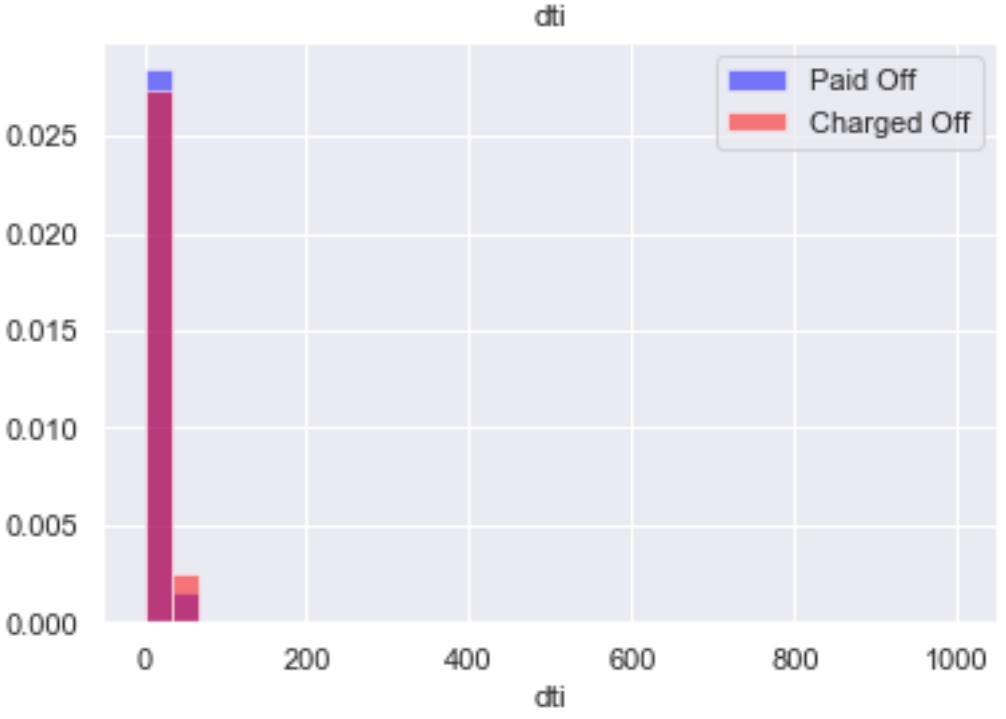

installment
Paid Off
Charged Off
0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
0
250
500
750
1000
1250
1500
1750
installment

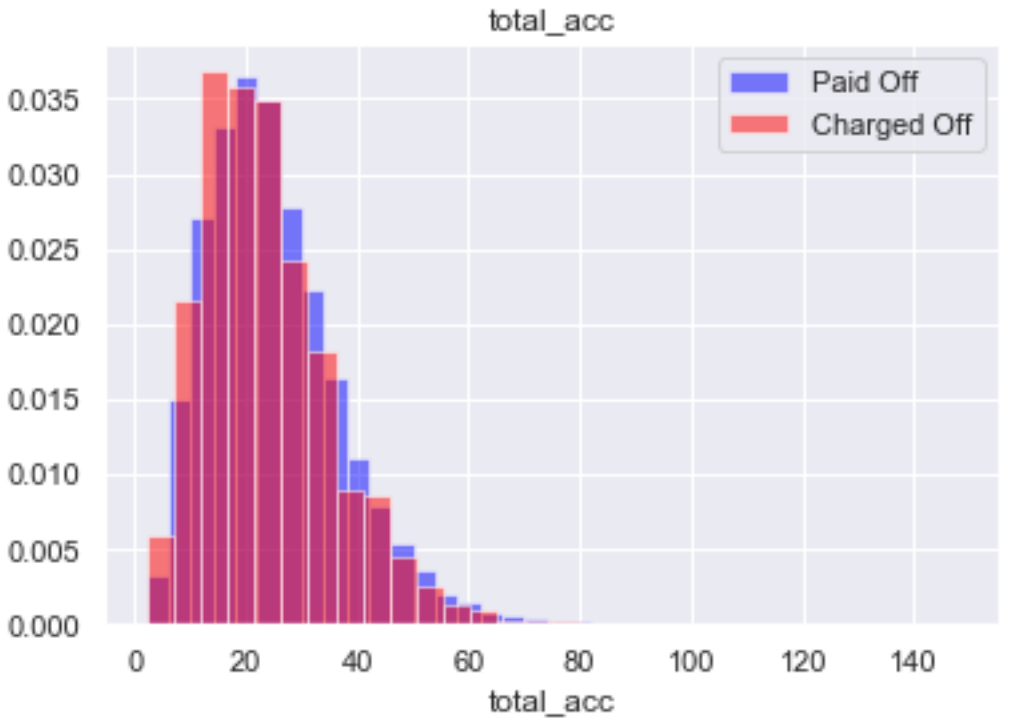

mo_sin_old_rev_tl_op
Paid Off
Charged Off
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0
100
200
300
400
500
600
700
mo_sin_old_rev_tl_op

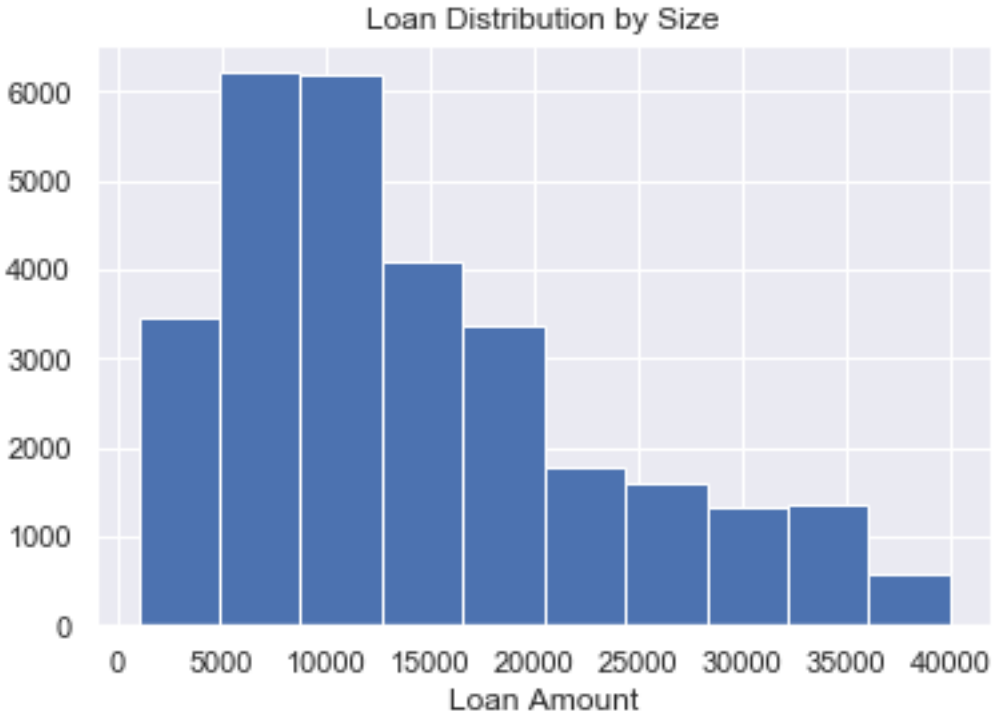

open_acc
Paid Off
Charged Off
0.08
0.06
0.04
0.02
0.00
0
10
20
30
40
50
60
70
open_acc


pub_rec
Paid Off
Charged Off
2.0
1.5
1.0
0.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
pub_rec

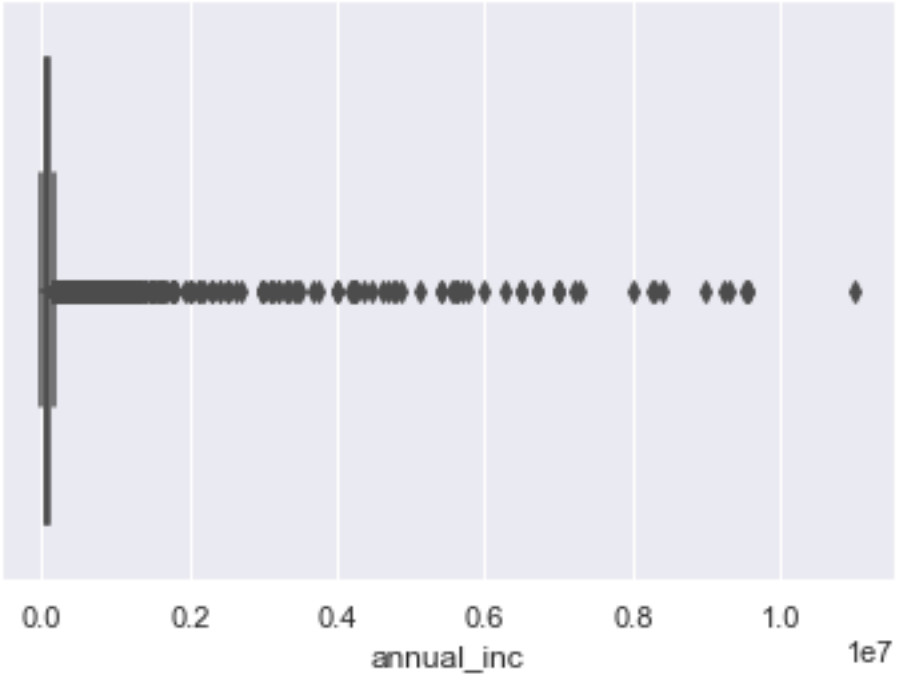

revol_bal
Paid Off
Charged Off
0.00004
0.00003
0.00002
0.00001
0.00000
0
100000
200000
300000
400000
500000
600000
revol_bal


dti
Paid Off
Charged Off
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000
dti


total_acc
Paid Off
Charged Off
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
120
140
total_acc

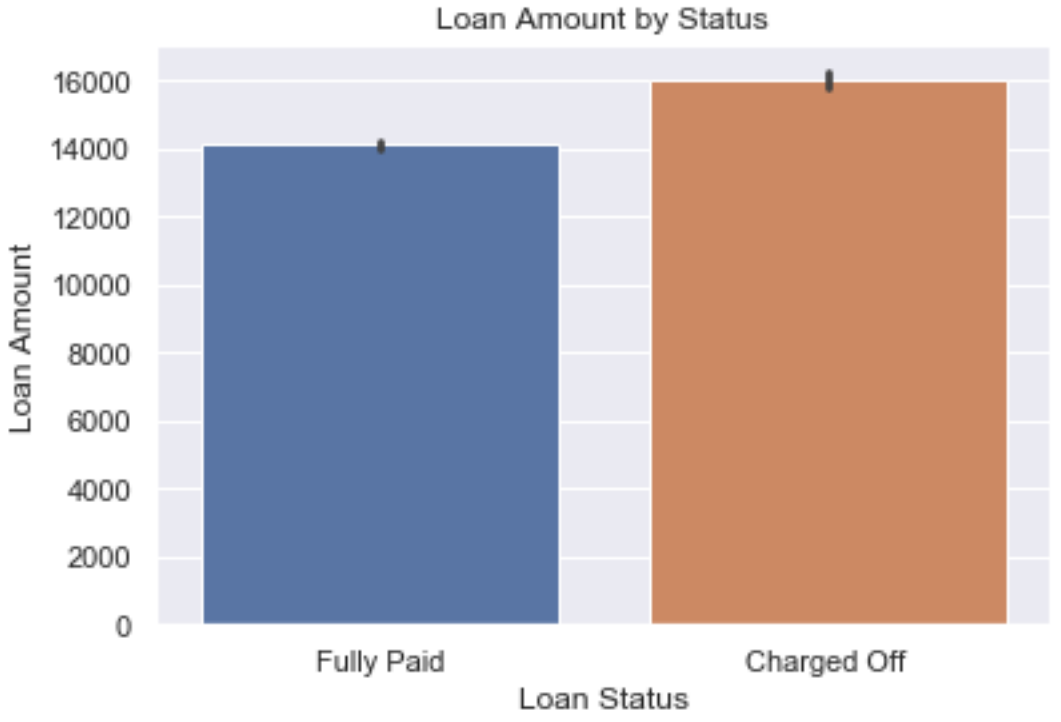

Loan Distribution by Size
6000
5000
4000
3000
2000
1000
0
0
5000
10000
15000
20000
25000
30000
35000
40000
Loan Amount


Annual Income relative to Loan Status
Annual Income
140000
120000
100000
80000
60000
40000
0
10000
20000
30000
40000
Loan Amount
loan_status
Fully Paid
Charged Off


0.0
0.2
0.4
0.6
0.8
1.0
1e7
annual_inc

Out[0]:

```
0           89600.0
1           70000.0
2           50000.0
3           75000.0
4          110000.0
             ...
142851     130000.0
142852     100000.0
142857      92000.0
142965      42000.0
142982     110000.0
Name: annual_inc, Length: 736424, dtype: float64
```



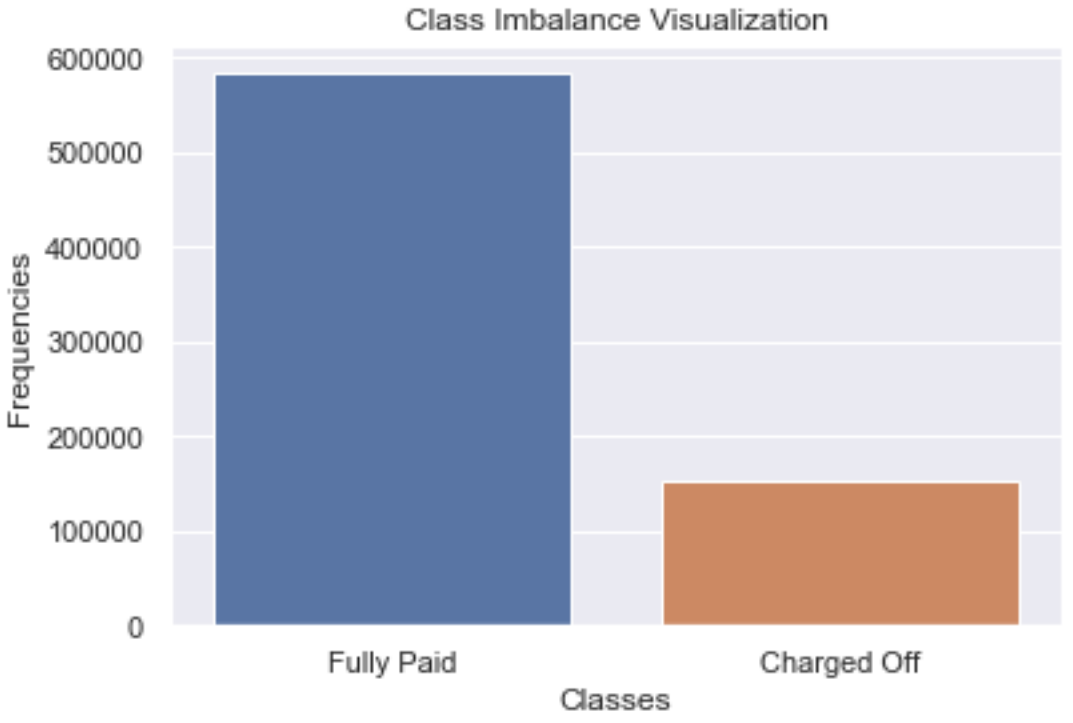
Target

The graph shows the classes are imbalanced since more loans will be Fully Paid versus Charged Off. Downsampling (Undersampling) was performed to randomly remove samples from the Fully Paid majority class.

Undersampling

Undersampling can be defined as removing some observations of the majority class. Undersampling can be a good choice when you have hundreads of thousands of rows. However this may remove some valuable information that may lead to underfitting and poor generalization to the test set.

Downsampling was used to reduce the number of loans Fully Paid from 437,623 to 114,695.



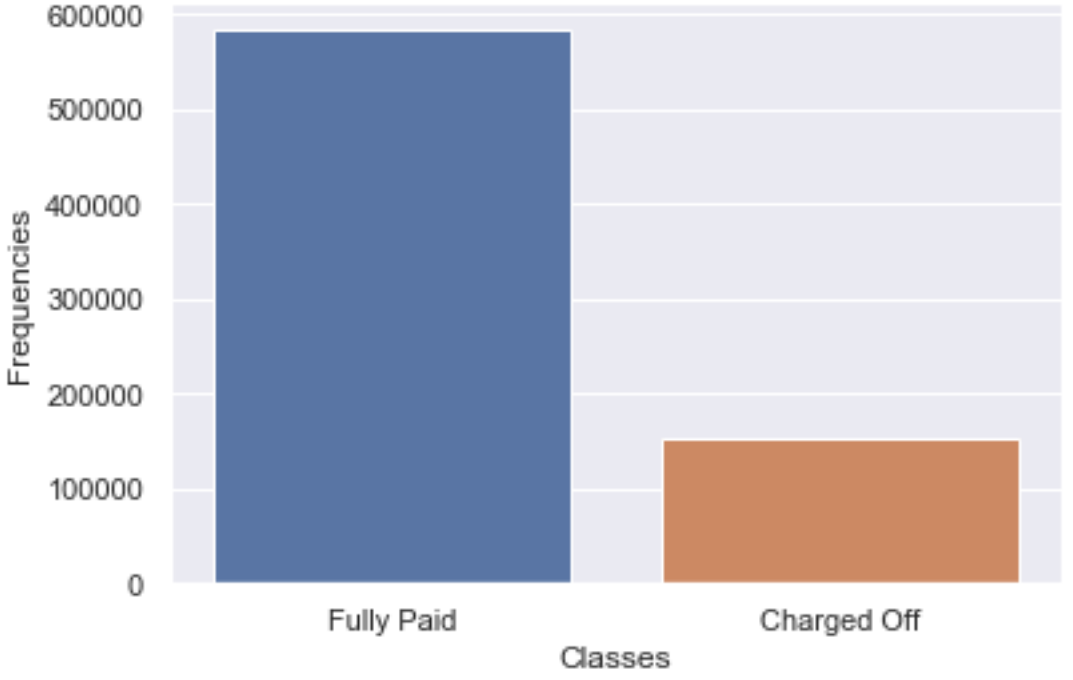
Clean Columns

Out[0]:

```
(736424, 64)
```

Out[0]:

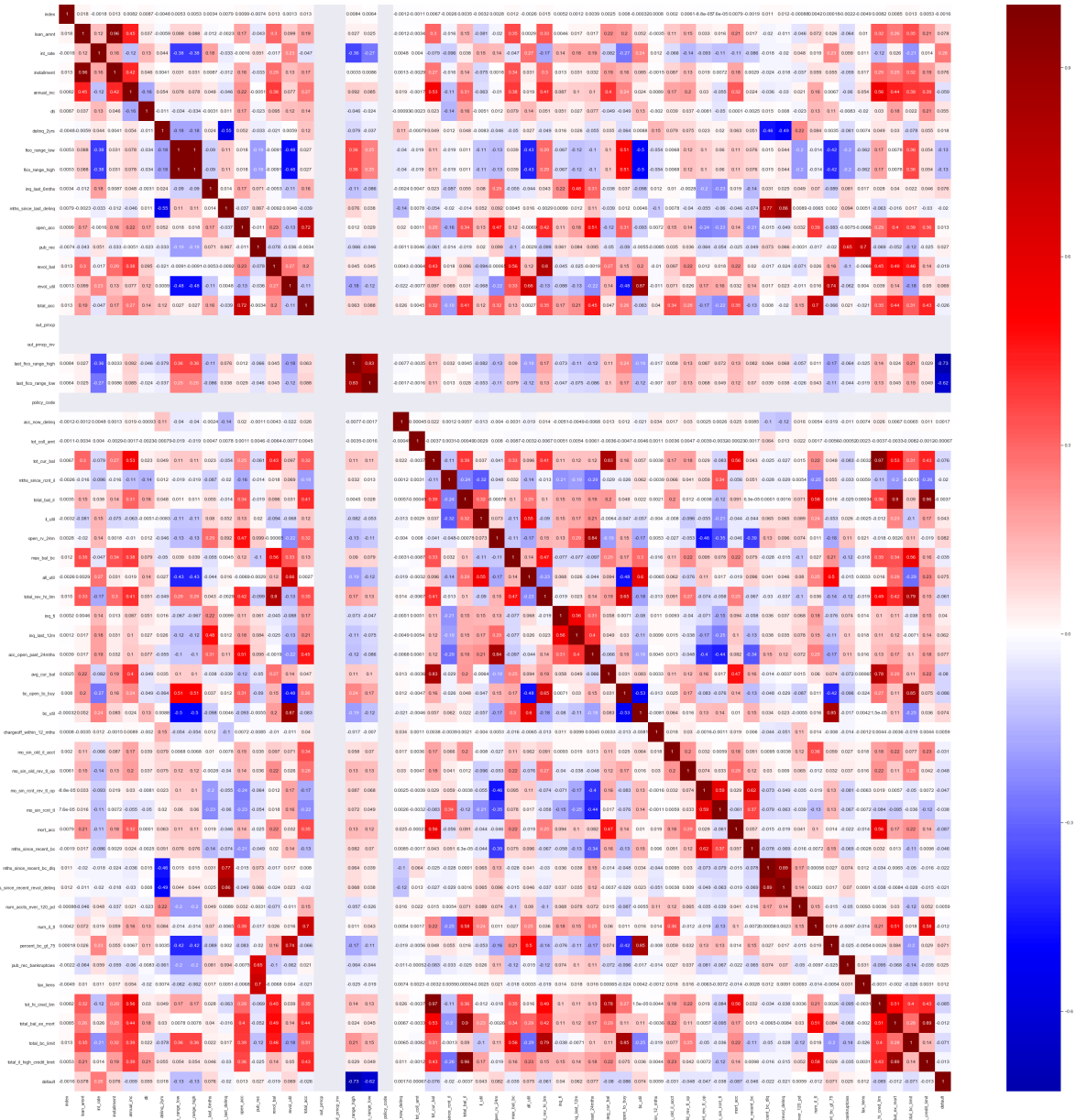
	index	loan_amnt	int_rate	installment	sub_grade	emp_length	home
0	0	30000.0	9.75%	964.50	B3	NaN	
1	1	25000.0	5.32%	752.87	A1	10+ years	
2	2	6950.0	7.39%	215.84	A4	2 years	
3	3	28000.0	11.47%	922.93	B5	10+ years	
4	4	7000.0	5.32%	210.81	A1	7 years	



Out[0]:

	loan_amnt	term	int_rate	installment	sub_grade	emp_length	hon
0	30000.0	36 months	0.0975	964.50	B3	NaN	
1	25000.0	36 months	0.0532	752.87	A1	10+ years	
2	6950.0	36 months	0.0739	215.84	A4	2 years	
3	28000.0	36 months	0.1147	922.93	B5	10+ years	
4	7000.0	36 months	0.0532	210.81	A1	7 years	

Feature Engineering



Out[0]:

```
B5    51453
C1    50099
B4    45190
C2    43236
C4    42309
B1    41554
C3    41205
C5    41172
B3    40951
B2    39724
A5    31409
A1    31308
A4    28966
D1    25686
D2    24703
A3    22925
A2    22591
D3    21050
D4    17374
D5    14802
E1     9841
E2     8285
E5     7747
E3     7744
E4     6846
F1     4229
F2     2771
F3     2429
F4     2035
F5     1860
G1     1566
G2      984
G3      835
G4      801
G5      744
Name: sub_grade, dtype: int64
```

```
[Index(['term', 'sub_grade', 'emp_length', 'home_own
ership',
       'verification_status', 'loan_status', 'purpos
e'],
      dtype='object')]
```

Out[0]:

```
1.000000    37107
0.376888     2849
1.015106     2530
0.374063     2527
1.022489     2354
            ...
0.630542        1
0.689781        1
0.681490        1
0.635036        1
0.722124        1
Name: fico_change, Length: 2517, dtype: int64
```

Out[0]:

	loan_amnt	term	int_rate	installment	annual_inc	loan_status	c
0	30000.0	36 months	0.0975	964.50	89600.0	0	21.1
1	25000.0	36 months	0.0532	752.87	70000.0	0	21.8
2	6950.0	36 months	0.0739	215.84	50000.0	0	5.6
3	28000.0	36 months	0.1147	922.93	75000.0	1	16.9
4	7000.0	36 months	0.0532	210.81	110000.0	0	6.3

Classificiation Model Selection and Evaluation

The selection is a progression from logistic, decision tree and finally to ensemble classification models.

Train and Test Accuracy scores were measured as well as the test precision and recall.

The goal was to minimize the number of false positives where a loan was classified as Charged Off.

Optimization of hyperparameters was performed on the selected models.

Modeling

```
loan_status
0    583498
1    152926
dtype: int64
```

Out[62]:

```
0    437623
1    114695
Name: loan_status, dtype: int64
```

Out[62]:

```
0    145875
1     38231
Name: loan_status, dtype: int64
```

Out[63]:

```
1    152926
0    114695
Name: loan_status, dtype: int64
```

Out[66]:

```
(Counter({0: 114695, 1: 152926}), Counter({0: 145875
, 1: 38231}))
```

Out[67]:

```
DummyClassifier(constant=None, random_state=None, st
rategy='most_frequent')
```

Out[67]:

```
0.20765754510988235
```

```
              precision    recall  f1-score   suppor
t

    Paid Off       0.00      0.00      0.00    14587
5
 Charged Off       0.21      1.00      0.34     3823
1

    accuracy                           0.21    18410
6
   macro avg       0.10      0.50      0.17    18410
6
weighted avg       0.04      0.21      0.07    18410
6
```

Random Forest

The RandomForestClassifier model was the first model. This model is helpful in classification problems where there is a clear class imbalance.

- Train Accuracy scores for the 5 folds: [0.91475688 0.91281387 0.9155519 0.91435404 0.9142544]
- Train Mean cross validation score: 0.914
- TrainStandard deviation of cross validatiion score: 0.001
- Test Accuracy score: 0.91
- Test: FBeta score favoring recall: 0.92

Out[0]:

```
RandomForestClassifier(bootstrap=True, class_weight=
None, criterion='gini',
                       max_depth=3, max_features='au
to', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, mi
n_impurity_split=None,
                       min_samples_leaf=1, min_sampl
es_split=2,
                       min_weight_fraction_leaf=0.0,
n_estimators=100,
                       n_jobs=None, oob_score=False,
random_state=25, verbose=0,
                       warm_start=False)


0.9132915578373895 0.9055218189521254
--- 20.08118748664856 seconds ---
```

Out[0]:

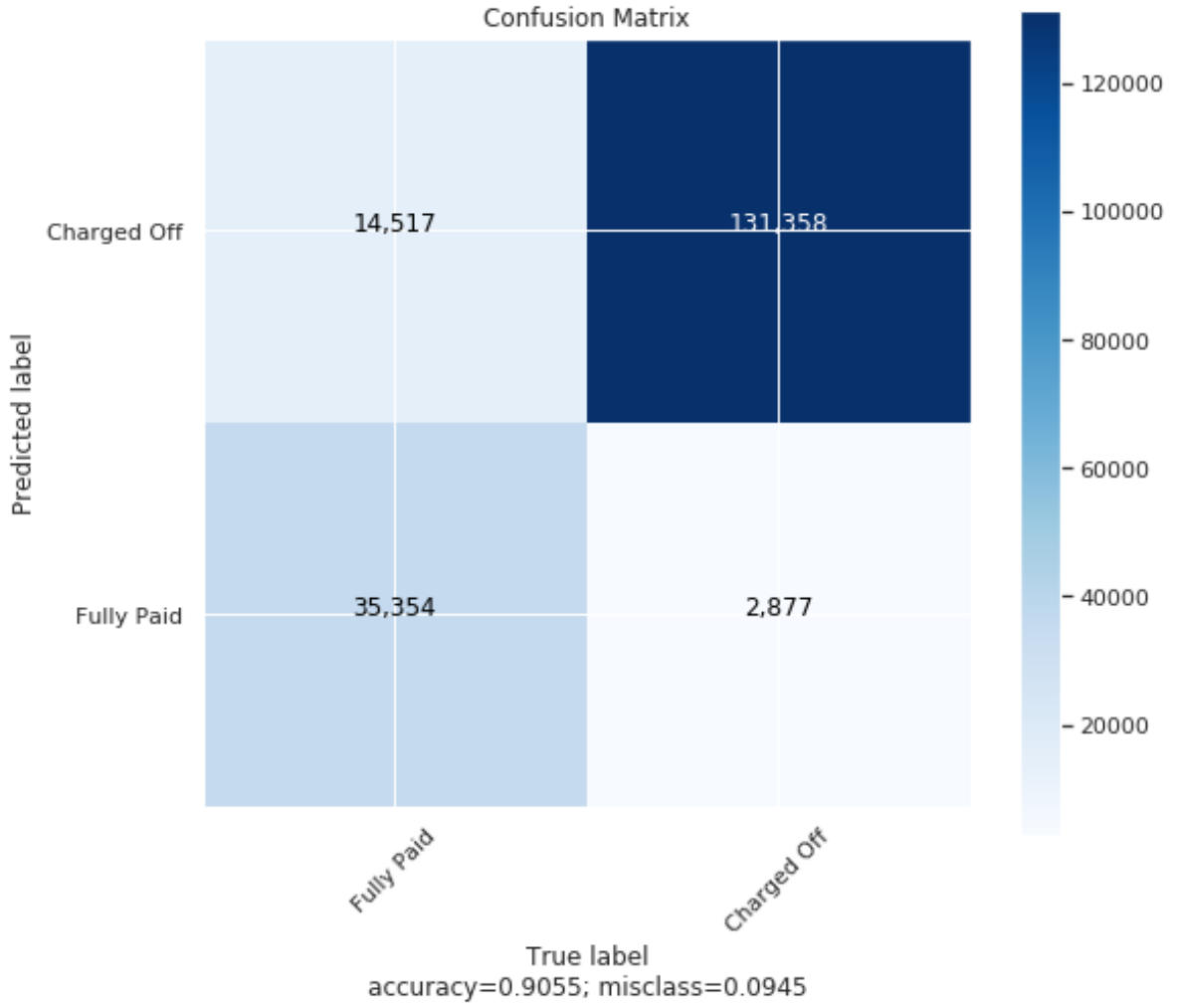
	Predicted_0	Predicted_1
Actual_0	131358	14517
Actual_1	2877	35354

Confusion Matrix Function

```
[[ 35354   2877]
 [ 14517 131358]]
```

Reports


```
[[ 35354   2877]
 [ 14517 131358]]
```



	precision	recall	f1-score	support
0	0.98	0.90	0.94	145875
1	0.71	0.92	0.80	38231
accuracy			0.91	184106
macro avg	0.84	0.91	0.87	184106
weighted avg	0.92	0.91	0.91	184106

Out[0]:

```
SelectFromModel(estimator=RandomForestClassifier(boo
tstrap=True,
                                                 cla
ss_weight=None,
                                                 cri
terion='gini', max_depth=3,
                                                 max
_features='auto',
                                                 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_e
stimators=100, n_jobs=None,
                                                 oob
_score=False,
                                                 ran
dom_state=25, verbose=0,
                                                 war
m_start=False),
                 max_features=None, norm_order=1, pre
fit=False, threshold=0.05)
```

```
['int_rate', 'last_fico_range_high', 'last_fico_rang
e_low', 'fico_change']
```

Out[0]:

```
RandomForestClassifier(bootstrap=True, class_weight=
None, criterion='entropy',
                       max_depth=5, max_features='au
to', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, mi
n_impurity_split=None,
                       min_samples_leaf=1, min_sampl
es_split=2,
                       min_weight_fraction_leaf=0.0,
n_estimators=10,
                       n_jobs=None, oob_score=False,
random_state=21, verbose=0,
                       warm_start=False)
```

Out[0]:

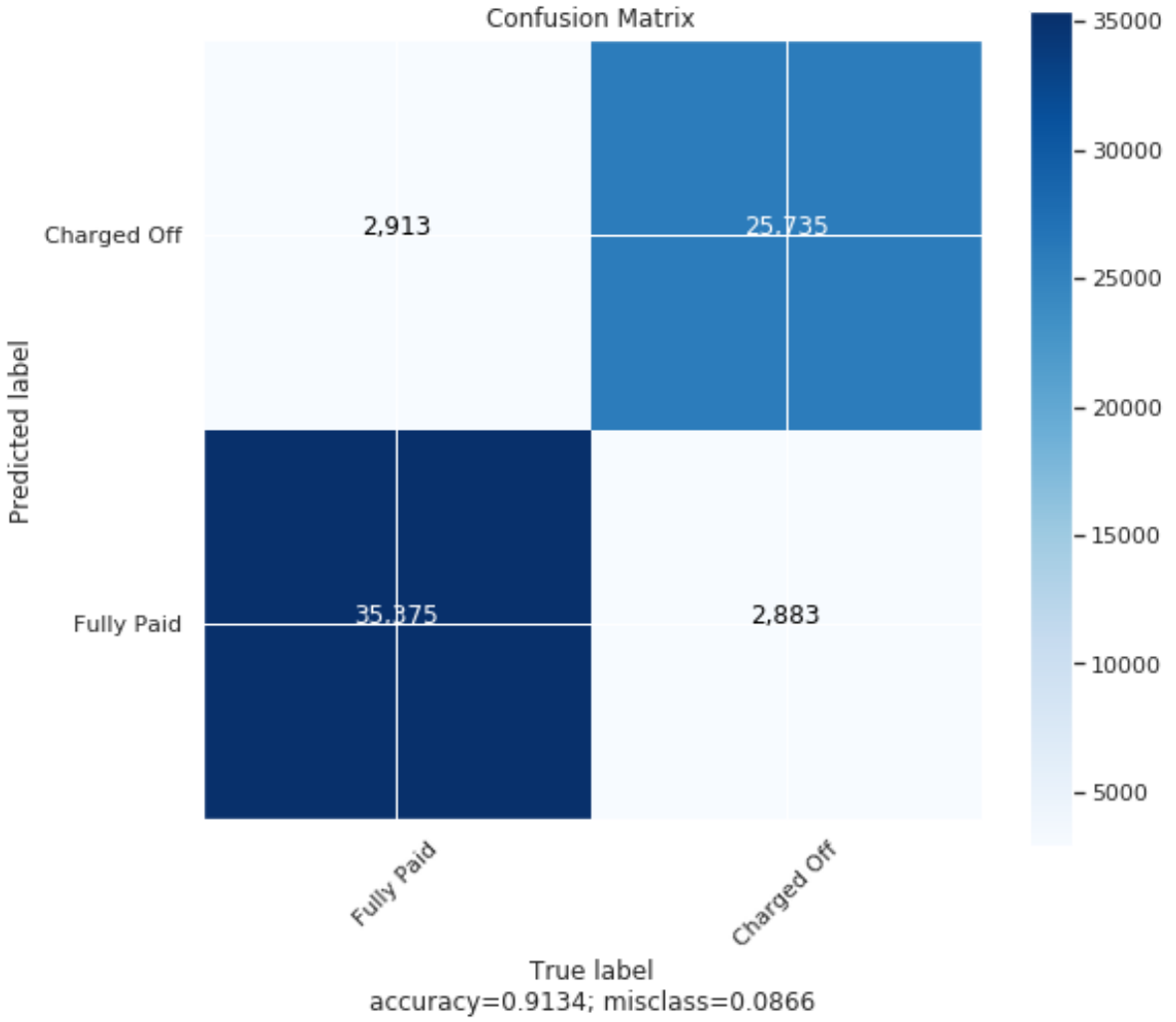
```
0.9133709981167608
```

Feature Importance

```
                          feature  importance
19             last_fico_range_low    0.331786
18            last_fico_range_high    0.251485
55                    fico_change    0.143211
2                        int_rate    0.059878
7                 fico_range_low    0.024535
8                fico_range_high    0.024480
34                   avg_cur_bal    0.021931
5                            dti    0.014921
3                    installment    0.011288
23                   tot_cur_bal    0.011040
105  verification_status_Verified    0.010705
1                      loan_amnt    0.010279
57                   sub_grade_A3    0.008642
14                    revol_util    0.007710
42                      mort_acc    0.007614
51                tot_hi_cred_lim    0.007428
35                 bc_open_to_buy    0.005831
56                   sub_grade_A2    0.005410
29                       all_util    0.004922
58                   sub_grade_A4    0.003989
```

Out[0]:

	Predicted_0	Predicted_1
Actual_0	25735	2913
Actual_1	2883	35375



	precision	recall	f1-score	support
0	0.90	0.90	0.90	28648
1	0.92	0.92	0.92	38258
accuracy			0.91	66906
macro avg	0.91	0.91	0.91	66906
weighted avg	0.91	0.91	0.91	66906

Out[0]:

[<matplotlib.lines.Line2D at 0x7f9805b6fd68>]

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f9805bbe710>]
```

Out[0]:

```
[<matplotlib.patches.Polygon at 0x7f9805b7b518>]
```

Out[0]:

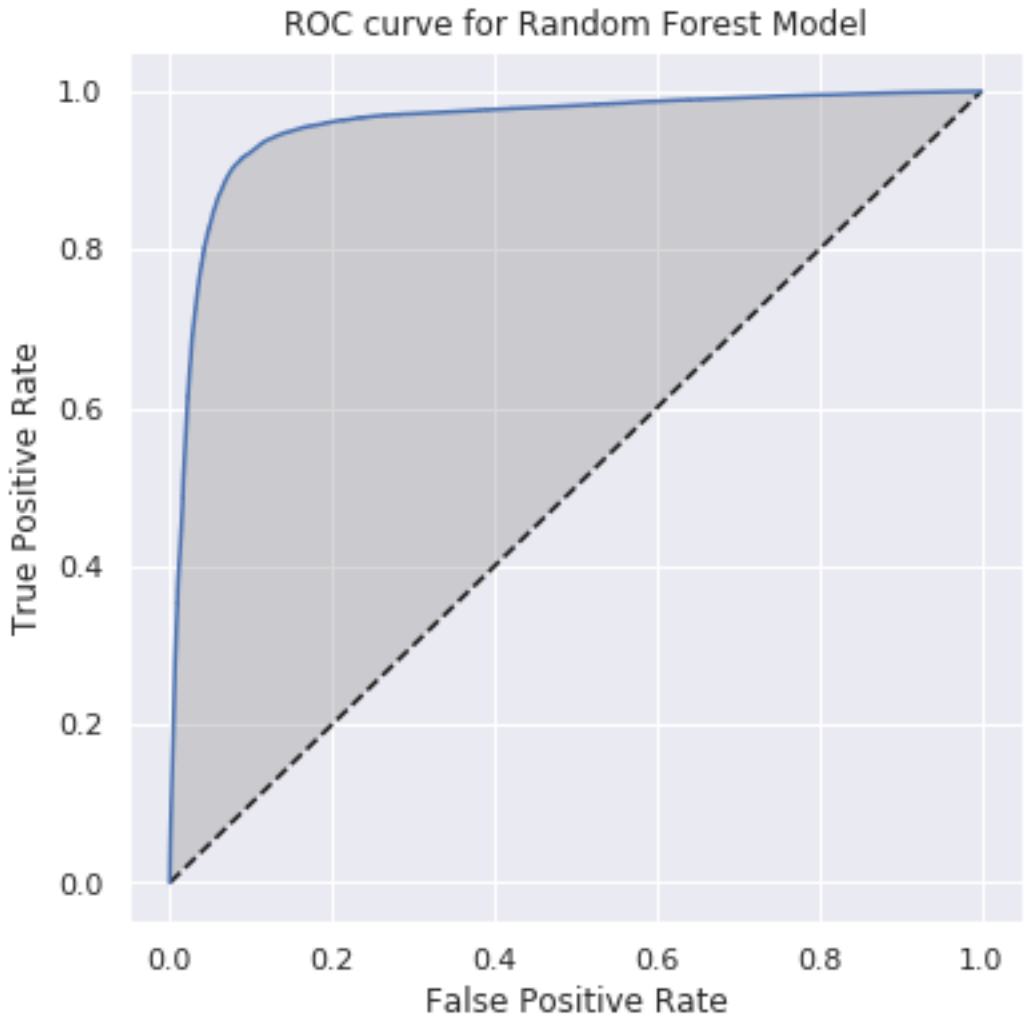
```
Text(0.5, 0, 'False Positive Rate')
```

Out[0]:

```
Text(0, 0.5, 'True Positive Rate')
```

Out[0]:

```
Text(0.5, 1.0, 'ROC curve for Random Forest Model')
```




```
Area under the ROC curve: 0.957
```

```
Accuracy scores for the 5 folds:  [0.91475688 0.9128
1387 0.9155519  0.91435404 0.9142544 ]
Mean cross validation score: 0.914
Standard deviation of cross validatiion score: 0.001
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f9805ae1048>]
```

Out[0]:

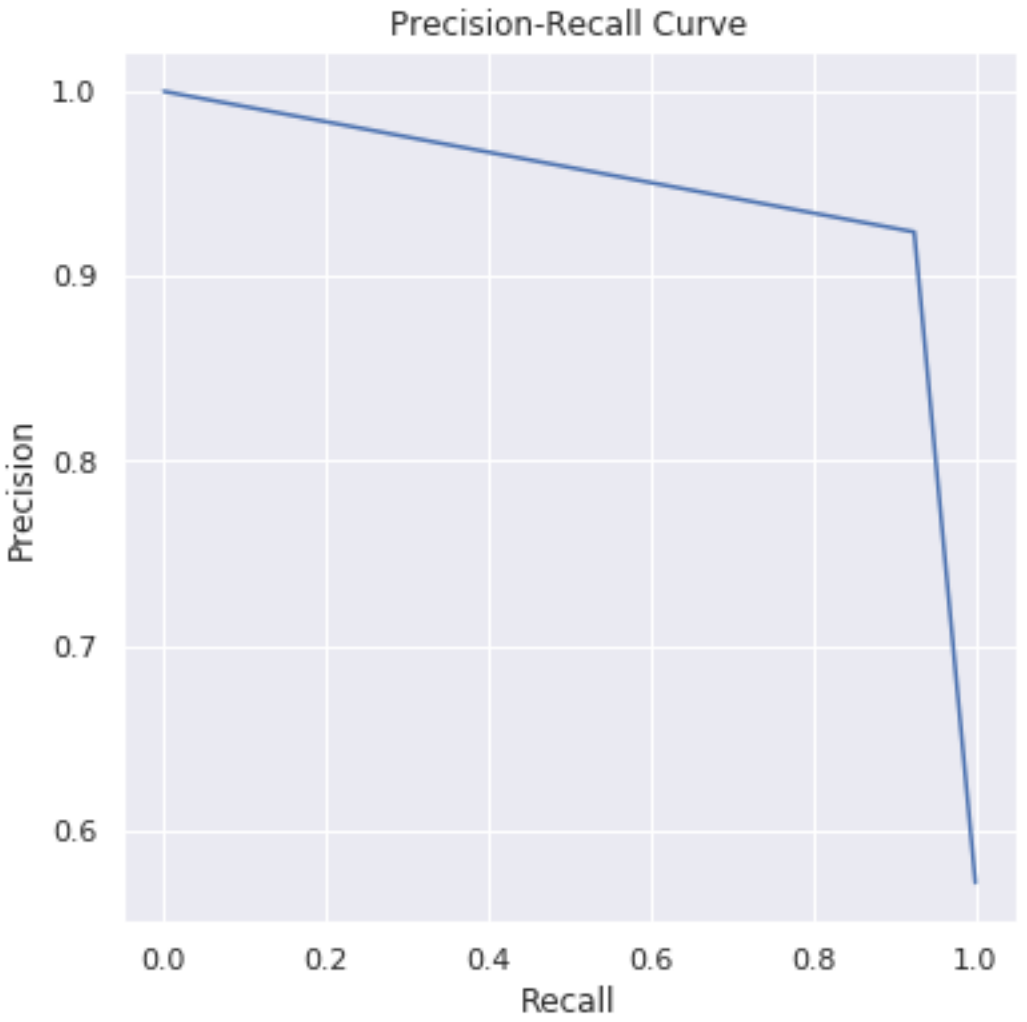
```
Text(0.5, 0, 'Recall')
```

Out[0]:

```
Text(0, 0.5, 'Precision')
```

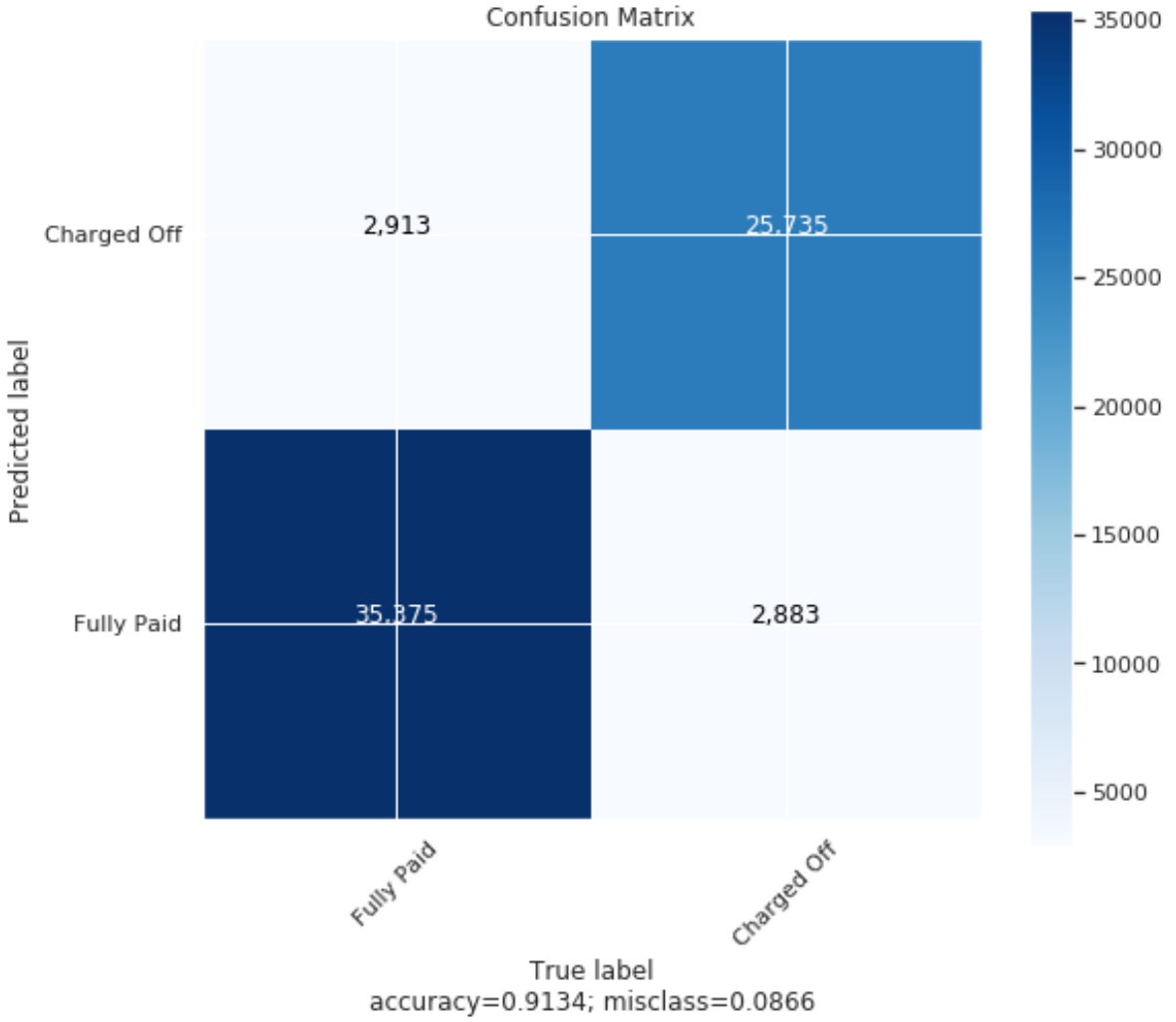
Out[0]:

```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



Out[0]:

```
array([[25735,  2913],
       [ 2883, 35375]])
```




```
Accuracy score: 0.91
Precision score: 0.92
Recall score: 0.92
F1 score: 0.92
Fbeta score favoring precision: 0.92
FBeta score favoring recall: 0.92
```

Logistic Regression

The second model was Logistic Regression. The initial parameters for the Logistic Regression were the default values and then I perfomed a GridSearch to find for the ideal C Value parameter.

- Train Accuracy scores for the 5 folds: [0.89649696 0.91869068 0.91908303 0.91949406 0.90867648]
- Train Mean cross validatiion score: 0.912
- Train Standard deviation of cross validatiion score: 0.009
- Test Accuracy score: 0.91
- Test FBeta score favoring recall: 0.88

Out[0]:

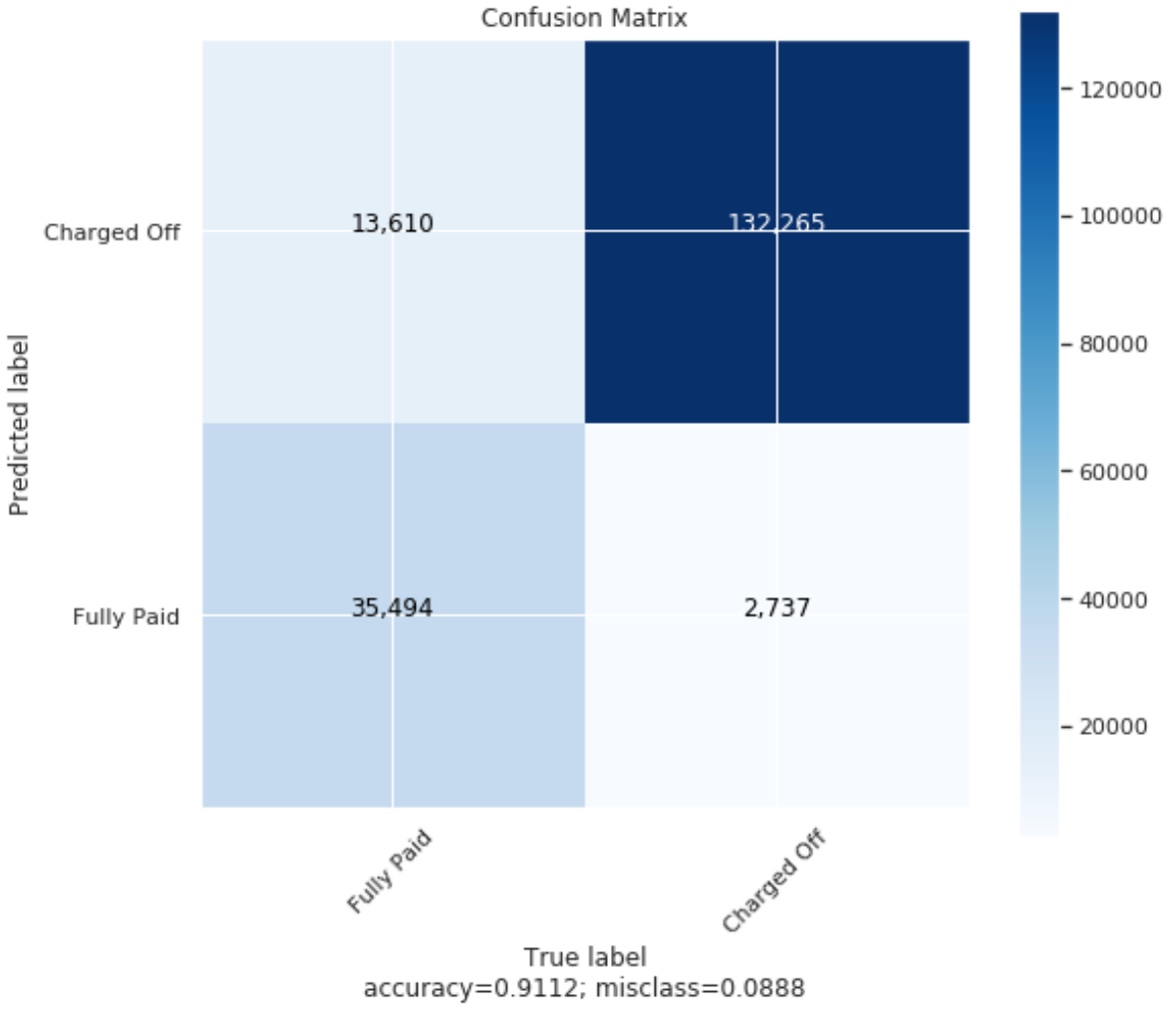
```
LogisticRegression(C=1, class_weight=None, dual=Fals
e, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=Non
e, max_iter=100,
                   multi_class='warn', n_jobs=None,
penalty='l2',
                   random_state=None, solver='warn',
tol=0.0001, verbose=0,
                   warm_start=False)

0.9182500625885114
0.9112087601707712


0.9112087601707712
CPU times: user 41.8 s, sys: 2.24 s, total: 44.1 s
Wall time: 3min 35s


{'C': 1}
```

Out[0]:

```
LogisticRegression(C=1.0, class_weight=None, dual=Fa
lse, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=Non
e, max_iter=100,
                   multi_class='warn', n_jobs=None,
penalty='l2',
                   random_state=None, solver='warn',
tol=0.0001, verbose=0,
                   warm_start=False)
```

Out[0]:

	Predicted_0	Predicted_1
Actual_0	132265	13610
Actual_1	2737	35494



```
Accuracy scores for the 5 folds:  [0.89649696 0.9186
9068 0.91910171 0.91949406 0.90869516]
Mean cross validatiion score: 0.912
Standard deviation of cross validatiion score: 0.009
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f8b9f6e7668>]
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f8b9f7a5b70>]
```

Out[0]:

```
[<matplotlib.patches.Polygon at 0x7f8b9f6e79e8>]
```

Out[0]:

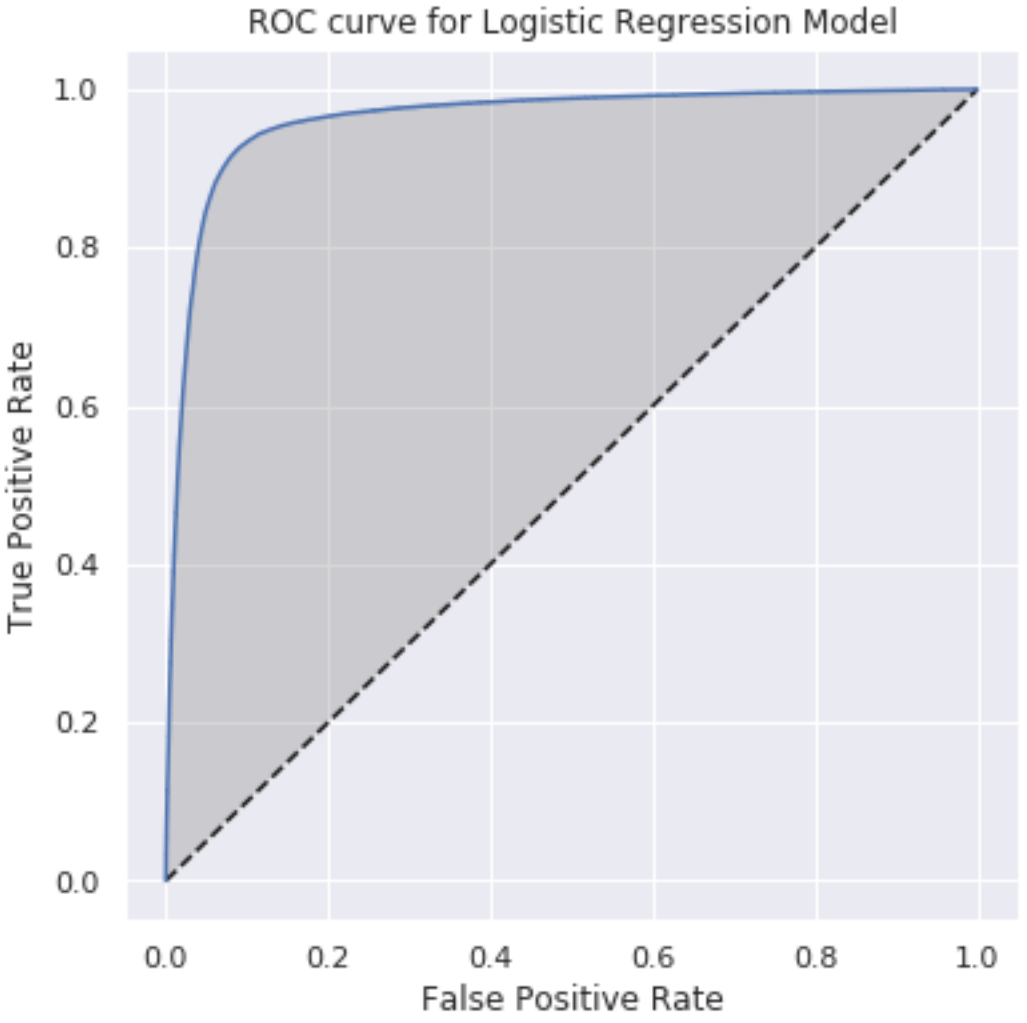
```
Text(0.5, 0, 'False Positive Rate')
```

Out[0]:

```
Text(0, 0.5, 'True Positive Rate')
```

Out[0]:

```
Text(0.5, 1.0, 'ROC curve for Logistic Regression Mo
del')
```




```
Area under the ROC curve: 0.962
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f8b9f6cb860>]
```

Out[0]:

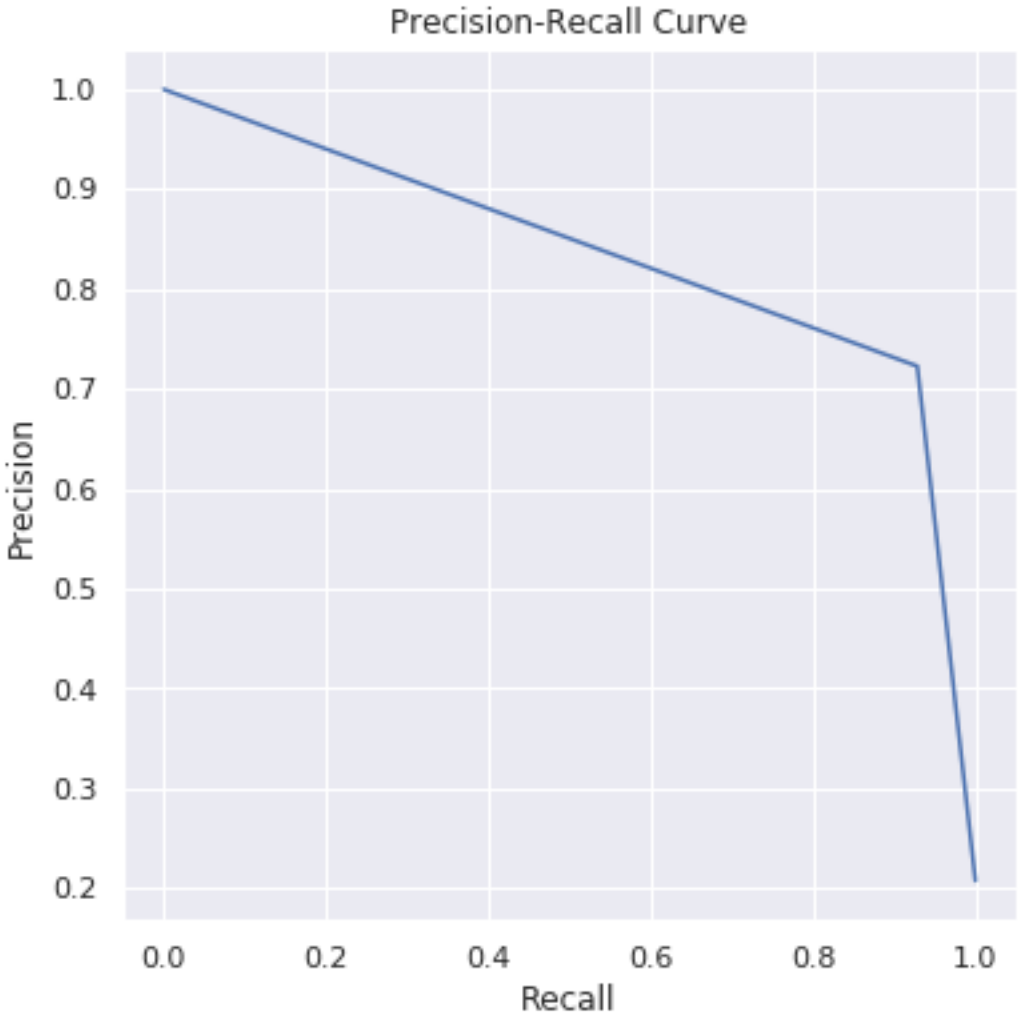
```
Text(0.5, 0, 'Recall')
```

Out[0]:

```
Text(0, 0.5, 'Precision')
```

Out[0]:

```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



```
Accuracy score: 0.91
Precision score: 0.72
Recall score: 0.93
F1 score: 0.81
Fbeta score favoring precision: 0.76
FBeta score favoring recall: 0.88
```

AdaBoost

The third model used was the Adaptive Boosting Classifier, which uses an ensemble of "weak learners" that produce a better predictive model. AdaBoost gives more weight to observations that are difficult to predict.

- Train Accuracy scores for the 5 folds: [0.9028865 0.91983036 0.92020402 0.9184478 0.9064345]
- Train Mean cross validatiion score: 0.914
- Train Standard deviation of cross validatiion score: 0.007
- Test FBeta score favoring recall: 0.93

Out[70]:

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimat
or=None, learning_rate=0.8,
                   n_estimators=100, random_state=No
ne)
```

```
0.9098399834877734
0.9170618150294633
```

```
              precision    recall  f1-score   suppor
t

           0       0.98      0.90      0.94    14587
5
           1       0.71      0.93      0.81     3823
1

    accuracy                           0.91    18410
6
   macro avg       0.85      0.92      0.87    18410
6
weighted avg       0.93      0.91      0.91    18410
6
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x1a6d3d5470>]
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x1a6d3d5198>]
```

Out[0]:

```
[<matplotlib.patches.Polygon at 0x1a6d3d5ba8>]
```

Out[0]:

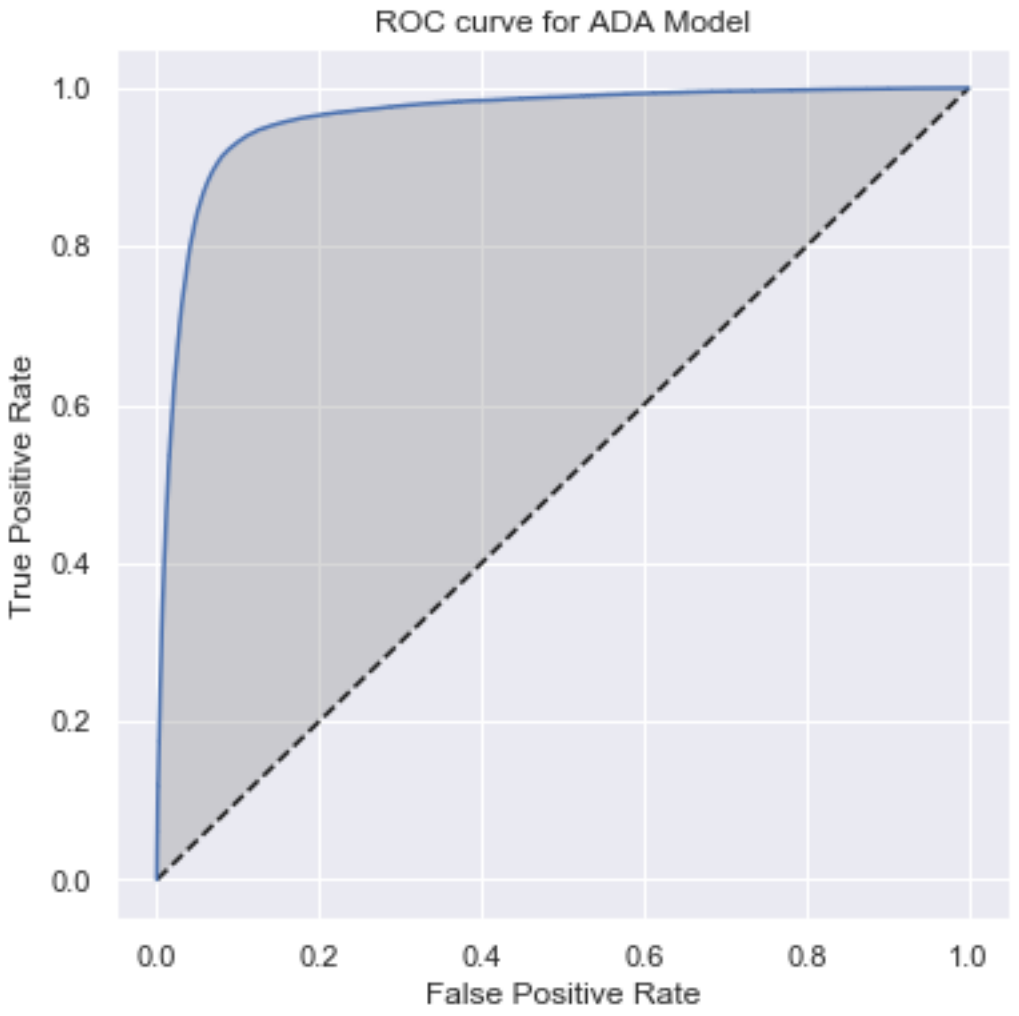
```
Text(0.5, 0, 'False Positive Rate')
```

Out[0]:

```
Text(0, 0.5, 'True Positive Rate')
```

Out[0]:

```
Text(0.5, 1.0, 'ROC curve for ADA Model')
```




```
Area under the ROC curve: 0.963
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x1b0ce3fa58>]
```

Out[0]:

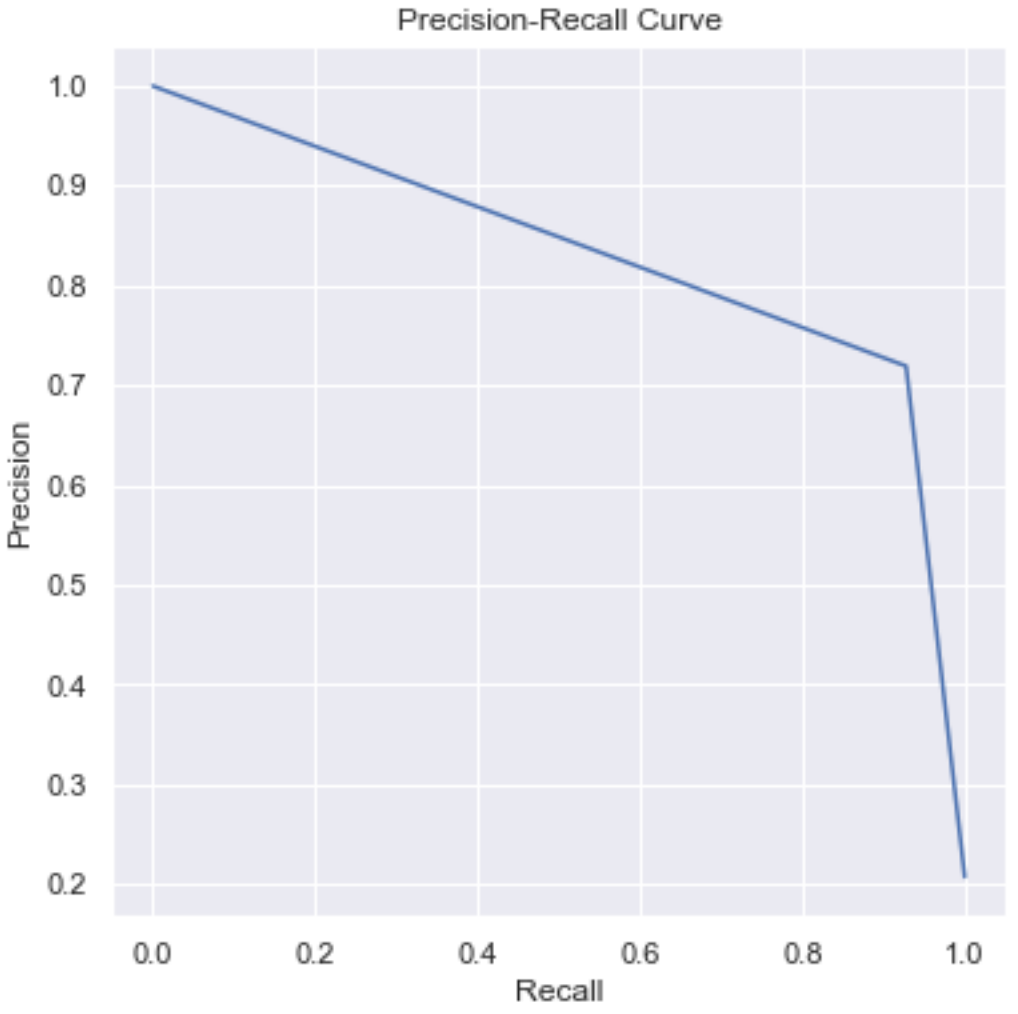
```
Text(0.5, 0, 'Recall')
```

Out[0]:

```
Text(0, 0.5, 'Precision')
```

Out[0]:

```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



Gradient Boosting

The fourth model used was the Gradient Boosting. A Gradient Boosted Classification model's goal is too minimize the loss function.

A GridSearch was performed to optimize the hyperparamaters.

- Train Accuracy scores for the 5 folds: [0.90326016 0.92059637 0.92173604 0.92050295 0.90916224]
- Train Mean cross validatiion score: 0.915
- Train Standard deviation of cross validatiion score: 0.007
- Test Accuracy score: 0.92
- Test FBeta score favoring recall: 0.93

Out[0]:

```
GradientBoostingClassifier(criterion='friedman_mse',
init=None,
                           learning_rate=0.1, loss='
deviance', max_depth=3,
                           max_features=None, max_le
af_nodes=None,
                           min_impurity_decrease=0.0
, min_impurity_split=None,
                           min_samples_leaf=1, min_s
amples_split=2,
                           min_weight_fraction_leaf=
0.0, n_estimators=100,
                           n_iter_no_change=None, pr
esort='auto',
                           random_state=None, subsam
ple=1.0, tol=0.0001,
                           validation_fraction=0.1,
verbose=0,
                           warm_start=False)
```

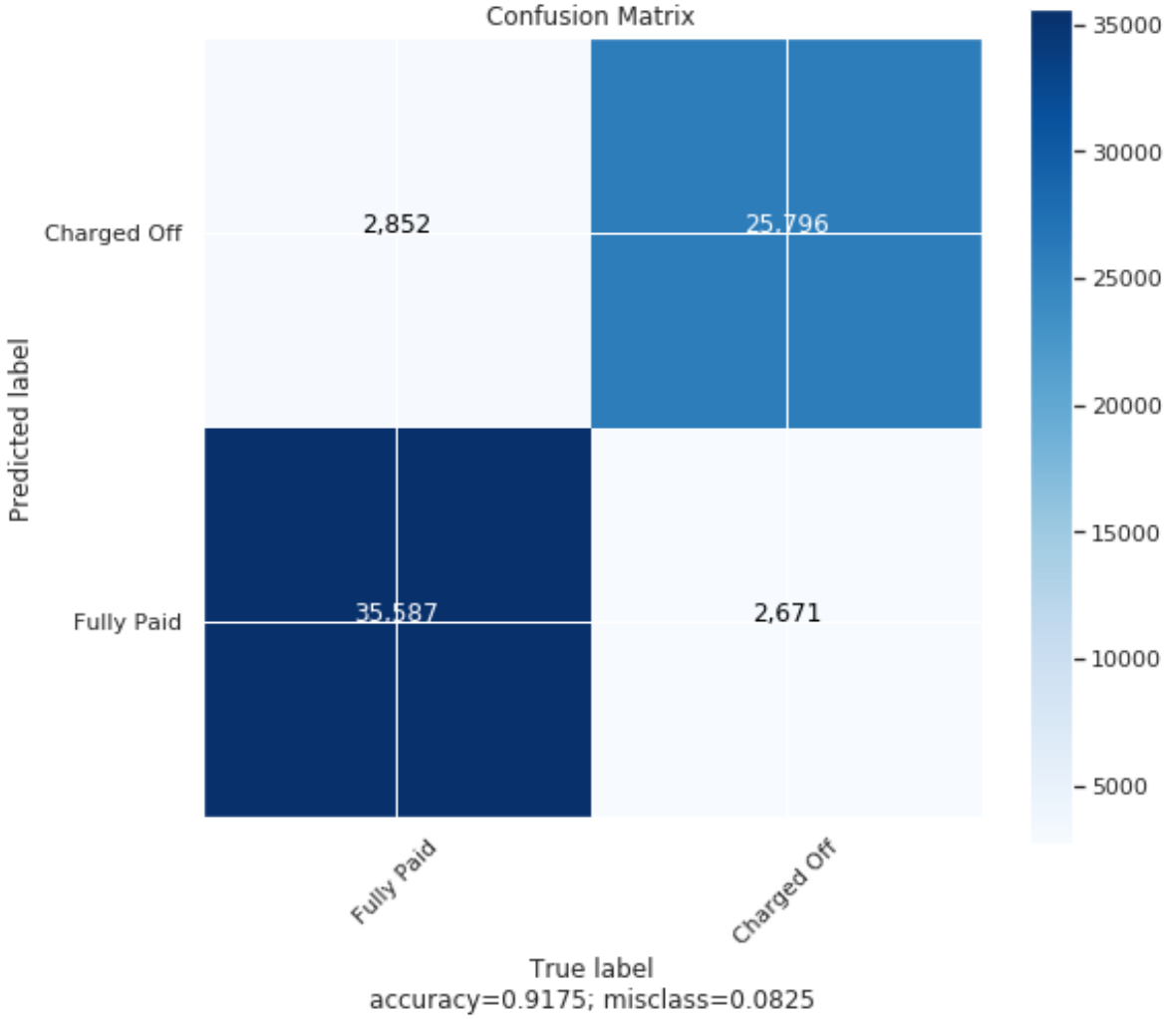
Out[0]:

```
(0.9188102533442941, 0.9174513496547395)
```

```
Accuracy scores for the 5 folds:  [0.90326016 0.9205
9637 0.92173604 0.92050295 0.90916224]
Mean cross validatiion score: 0.915
Standard deviation of cross validatiion score: 0.007
```

Out[0]:

	Predicted_0	Predicted_1
Actual_0	25796	2852
Actual_1	2671	35587



Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f9805967390>]
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f9805a78668>]
```

Out[0]:

```
[<matplotlib.patches.Polygon at 0x7f9805967710>]
```

Out[0]:

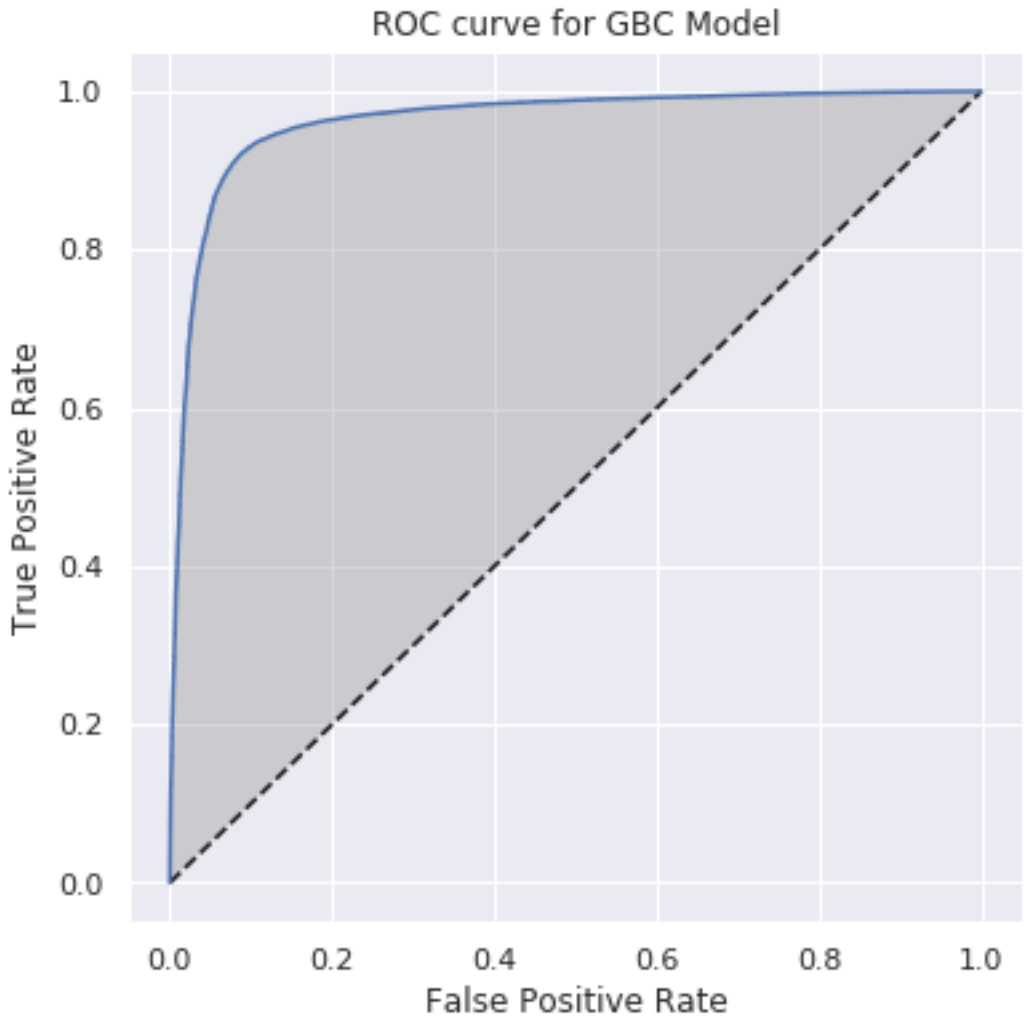
```
Text(0.5, 0, 'False Positive Rate')
```

Out[0]:

```
Text(0, 0.5, 'True Positive Rate')
```

Out[0]:

```
Text(0.5, 1.0, 'ROC curve for GBC Model')
```




```
Area under the ROC curve: 0.963
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f9805949588>]
```

Out[0]:

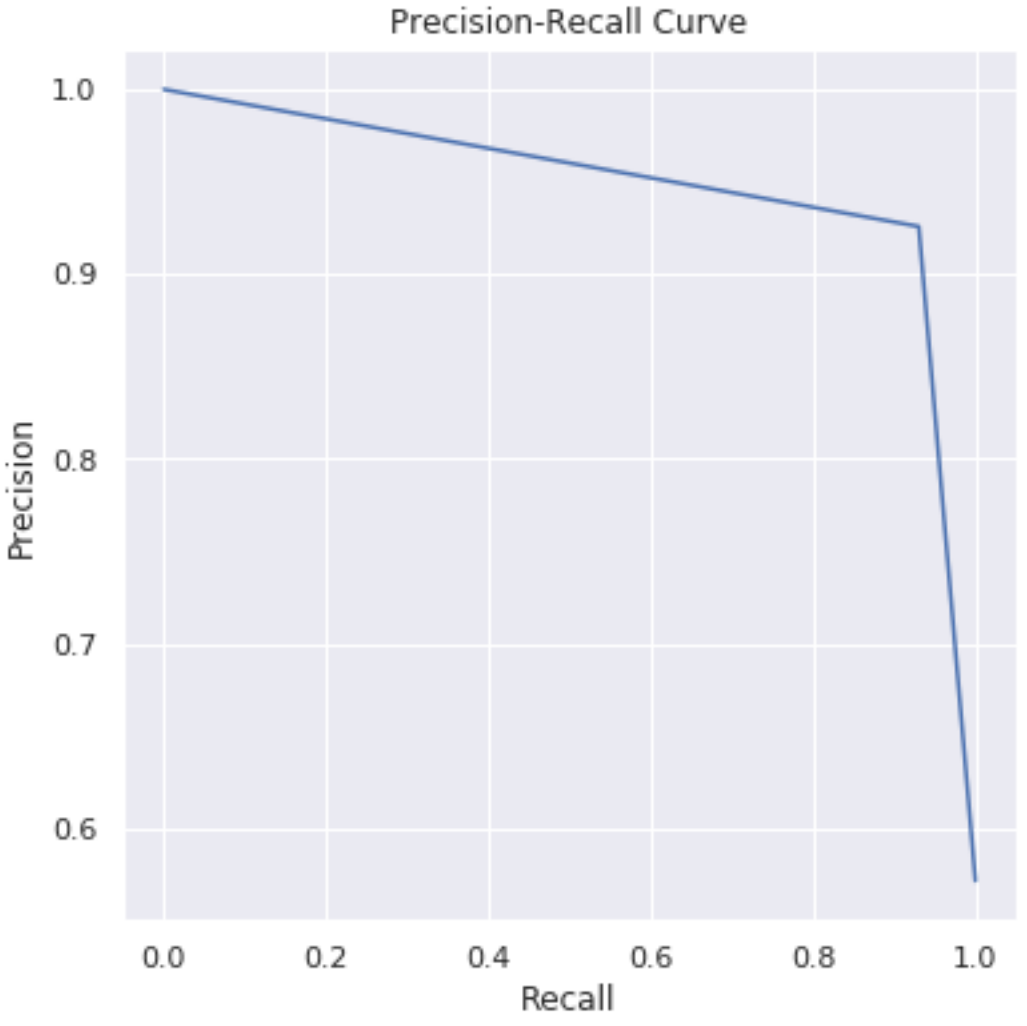
```
Text(0.5, 0, 'Recall')
```

Out[0]:

```
Text(0, 0.5, 'Precision')
```

Out[0]:

```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



```
Accuracy score: 0.92
Precision score: 0.93
Recall score: 0.93
F1 score: 0.93
Fbeta score favoring precision: 0.93
FBeta score favoring recall: 0.93
```

XGBoost

eXtreme Gradient Boosting. The Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

- Train Accuracy scores for the 5 folds: [0.9056142 0.92104476 0.92132501 0.92012929 0.90895673]
- Train Mean cross validatiion score: 0.915
- Train Standard deviation of cross validatiion score: 0.007
- Test Accuracy score: 0.92
- Test FBeta score favoring recall: 0.93

Out[0]:

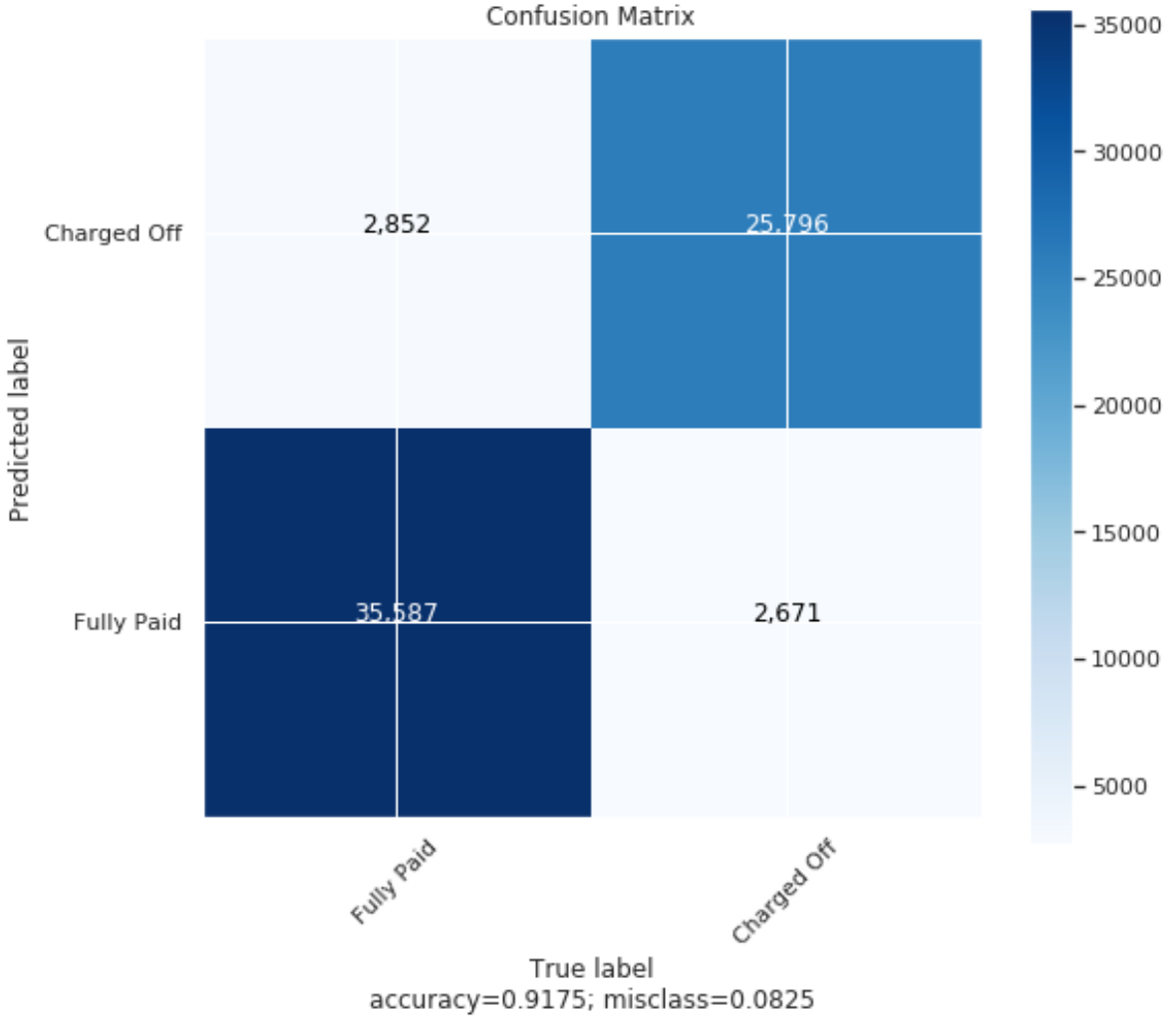
```
XGBClassifier(base_score=0.5, booster='gbtree', cols
ample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1
, gamma=0,
              learning_rate=0.1, max_delta_step=0, m
ax_depth=3,
              min_child_weight=1, missing=None, n_es
timators=100, n_jobs=1,
              nthread=None, objective='binary:logist
ic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_w
eight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

Out[0]:

```
0.9173467252563298
```

Out[0]:

	Predicted_0	Predicted_1
Actual_0	25785	2863
Actual_1	2667	35591




```
Accuracy scores for the 5 folds:  [0.9056142  0.9210
4476 0.92132501 0.92012929 0.90895673]
Mean cross validatiion score: 0.915
Standard deviation of cross validatiion score: 0.007
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f97fa6f9320>]
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f97fa6c8128>]
```

Out[0]:

```
[<matplotlib.patches.Polygon at 0x7f97fa6f96a0>]
```

Out[0]:

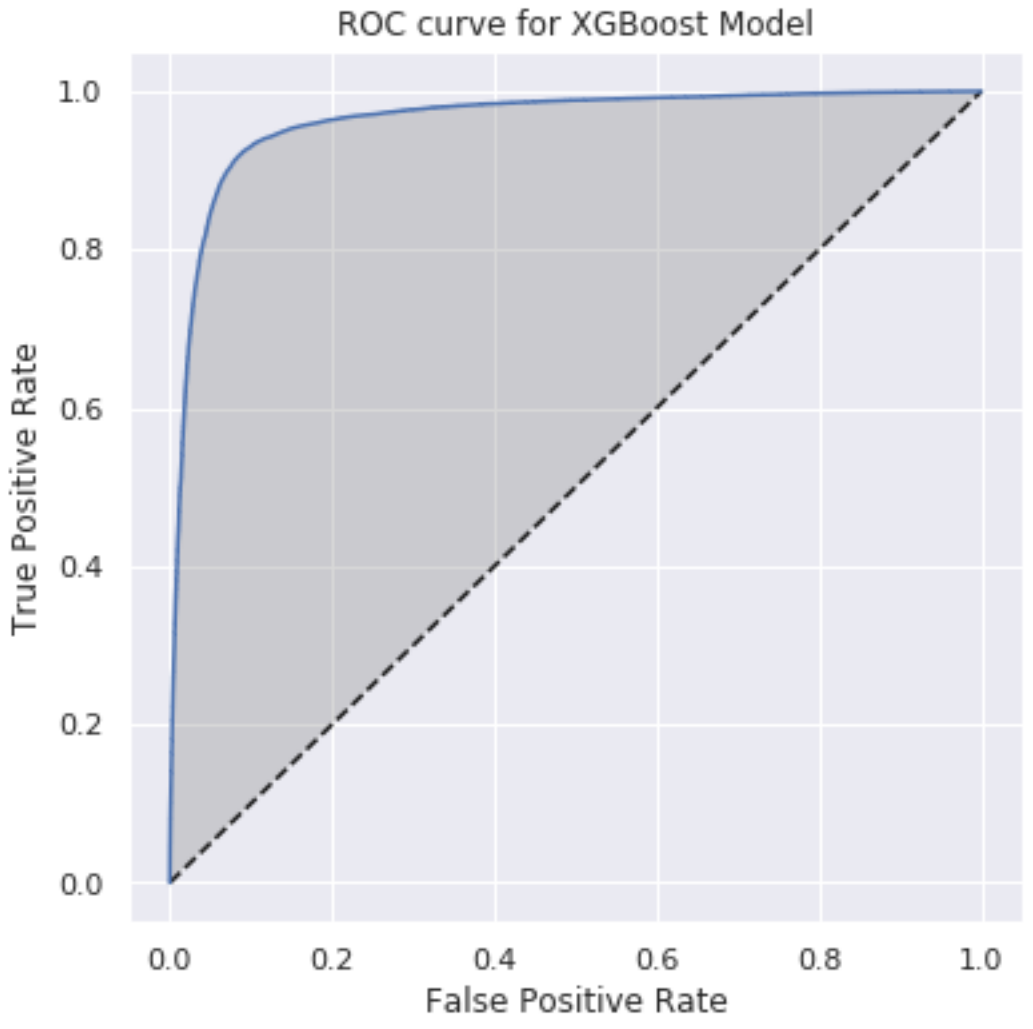
```
Text(0.5, 0, 'False Positive Rate')
```

Out[0]:

```
Text(0, 0.5, 'True Positive Rate')
```

Out[0]:

```
Text(0.5, 1.0, 'ROC curve for XGBoost Model')
```




```
Area under the ROC curve: 0.963
```

Out[0]:

```
[<matplotlib.lines.Line2D at 0x7f97fa65c518>]
```

Out[0]:

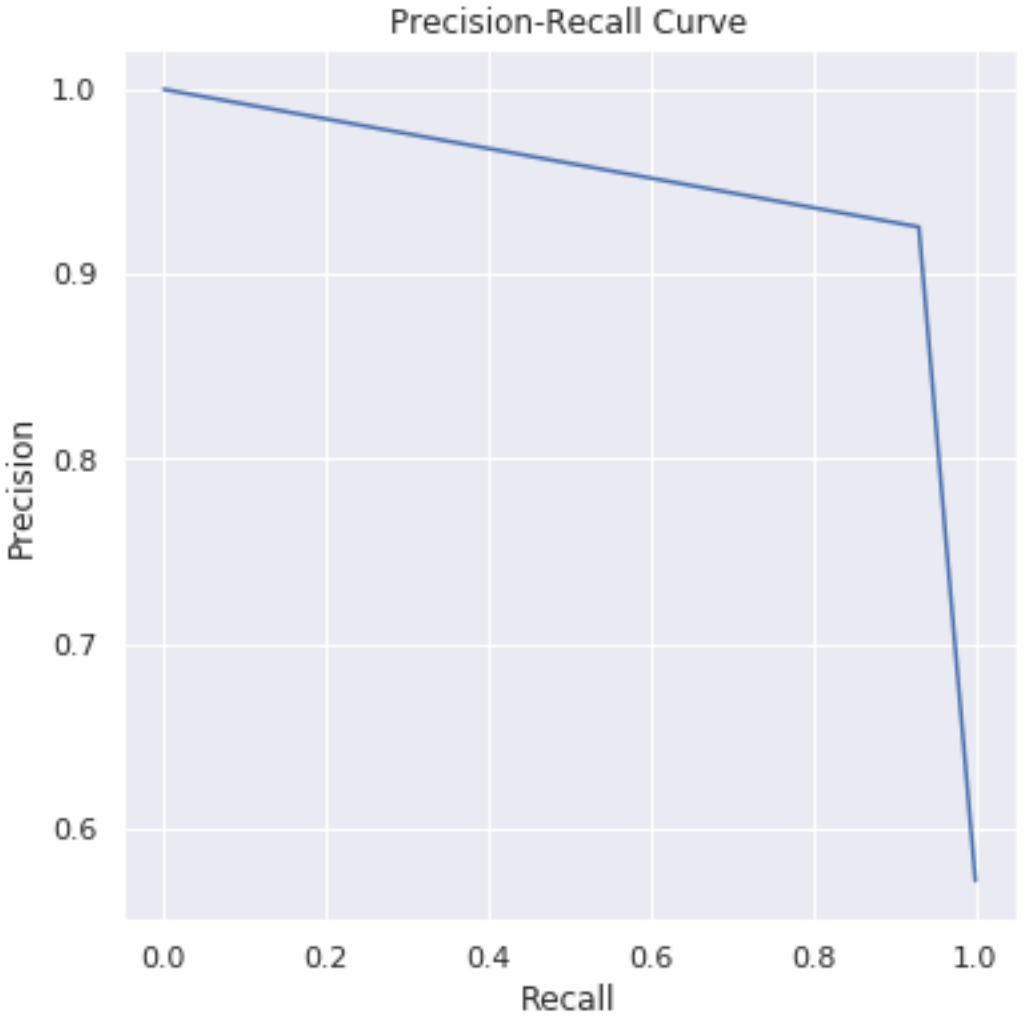
```
Text(0.5, 0, 'Recall')
```

Out[0]:

```
Text(0, 0.5, 'Precision')
```

Out[0]:

```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



```
Accuracy score: 0.92
Precision score: 0.93
Recall score: 0.93
F1 score: 0.93
Fbeta score favoring precision: 0.93
FBeta score favoring recall: 0.93


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 736424 entries, 0 to 736423
Columns: 120 entries, index to purpose_wedding
dtypes: float64(55), int64(2), uint8(63)
memory usage: 364.5 MB
```

Model Summary Scores

Model	Random Forest	Logistic	AdaBoost	GradientBoost	XGBoost
Mean Train Score	0.914	0.912	0.914	**0.915**	**0.915**
Train STD	0.001	0.009	0.007	**0.007**	**0.007**
Test Accuracy Score	0.910	0.910	0.920	**0.920**	**0.920**
FBeta score favoring recall	0.920	0.88	0.930	**0.930**	**0.930**

Summary

Based on the precision and recall scores of the model, we are able to match the current weighted average charge off rate on LendingClub's loans of 8.20%.

This lends credence to the company using a similar proprietary scoring model based on the features used in this project.

Gradient Boosting and eXtreme Gradient Boosting Train CV Score, Test Accuracy Score, and Test Fbeta score favoring recall were identical and either can be used for future development.

Next Steps

Lending Club has done a great job at optimizing their business model through the migration of loans from 60 month to 36 months. This has decreased the default rates, the duration of their investment portfolio to 16 months as well increased the issuance of loans on their marketplace.

Future development will now focus on external factors incorporated into the existing dataset and tailoring the ability to predict charge offs based on more select subsample criteria.

External data such as consumer credit change, demographic data, unemployment data, word analysis of the Federal Reserve forward guidance, and retails sales for a better view on lending as that is the key driver to the business.