Bigdata Project Report

June 8, 2021

```
                  Team #1
+--------------------------------+-----+----+
|              Name              | SEC | BN |
+--------------------------------+-----+----+
| Abdulrahman Khalid Hassan      |   1 | 30 |
| Mahmoud Othman Adas            |   2 | 19 |
| Yosry Mohamed Yosry            |   2 | 33 |
| Ahmad Mahmoud AbdElMen'em Afifi |   1 |  5 |
+--------------------------------+-----+----+
```

1 Introduction

LendingClub is a US peer-to-peer lending company, headquartered in San Francisco, California. It was the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission, and to offer loan trading on a secondary market.

LendingClub is the world's largest peer-to-peer lending platform, Given historical data on loans given out with information on whether or not the borrower defaulted (charge-off), we can build a model that can predict if a borrower will pay back their loan. This way in the future when we get a new potential customer, we can assess if they are likely to pay back the loan.

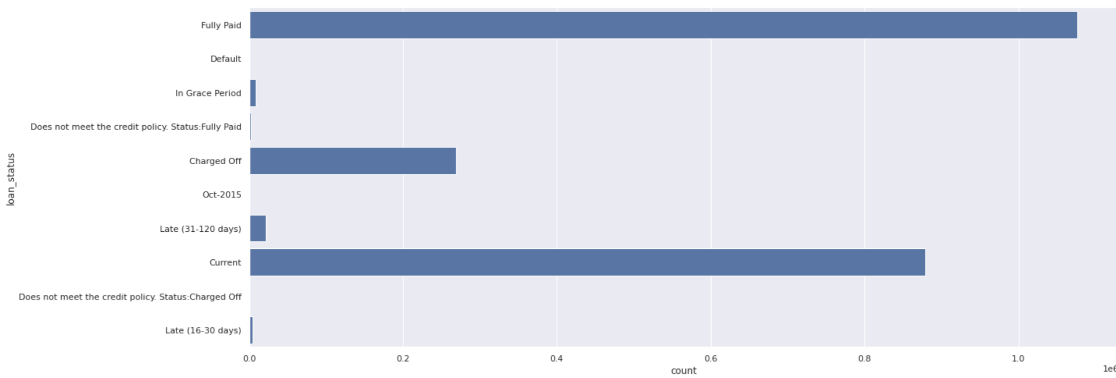
Objectives of this notebook is: - To show step-by-step how to visualize the dataset. - Data cleaning and preprocessing. - Assess whether or not a new customer is likely to pay back the loan.

Univariant Visualization

1.1 Loan Status Distribution

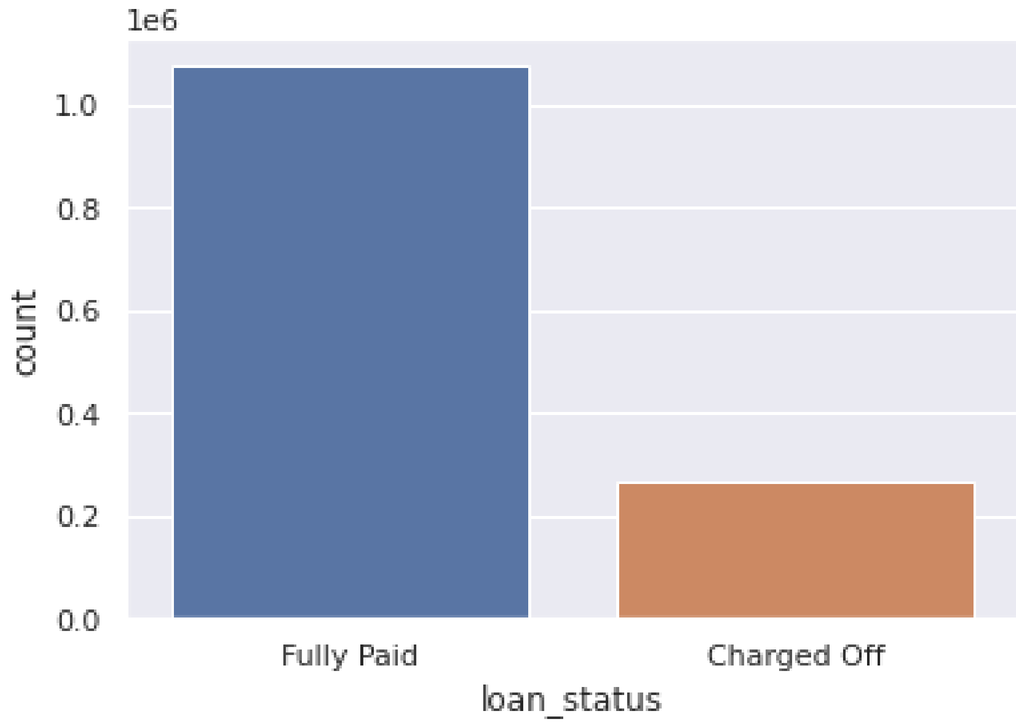
```
+--------------------+-------+
|         loan_status|  count|
+--------------------+-------+
|          Fully Paid|1076751|
|             Default|     40|
|                null|     33|
|     In Grace Period|   8436|
|Does not meet the...|   1988|
```

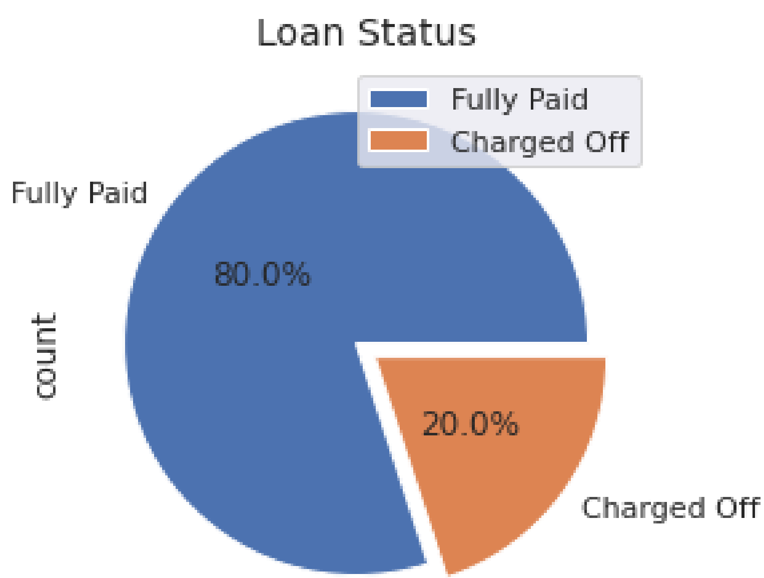
```
|         Charged Off| 268558|
|            Oct-2015|      1|
|  Late (31-120 days)|  21467|
|             Current| 878317|
|Does not meet the...|    761|
|   Late (16-30 days)|   4349|
+--------------------+-------+
```



filter the loan status to be only fully paid and charged off

```
+-----------+-------+
|loan_status|  count|
+-----------+-------+
| Fully Paid|1076751|
|Charged Off| 268558|
+-----------+-------+
```

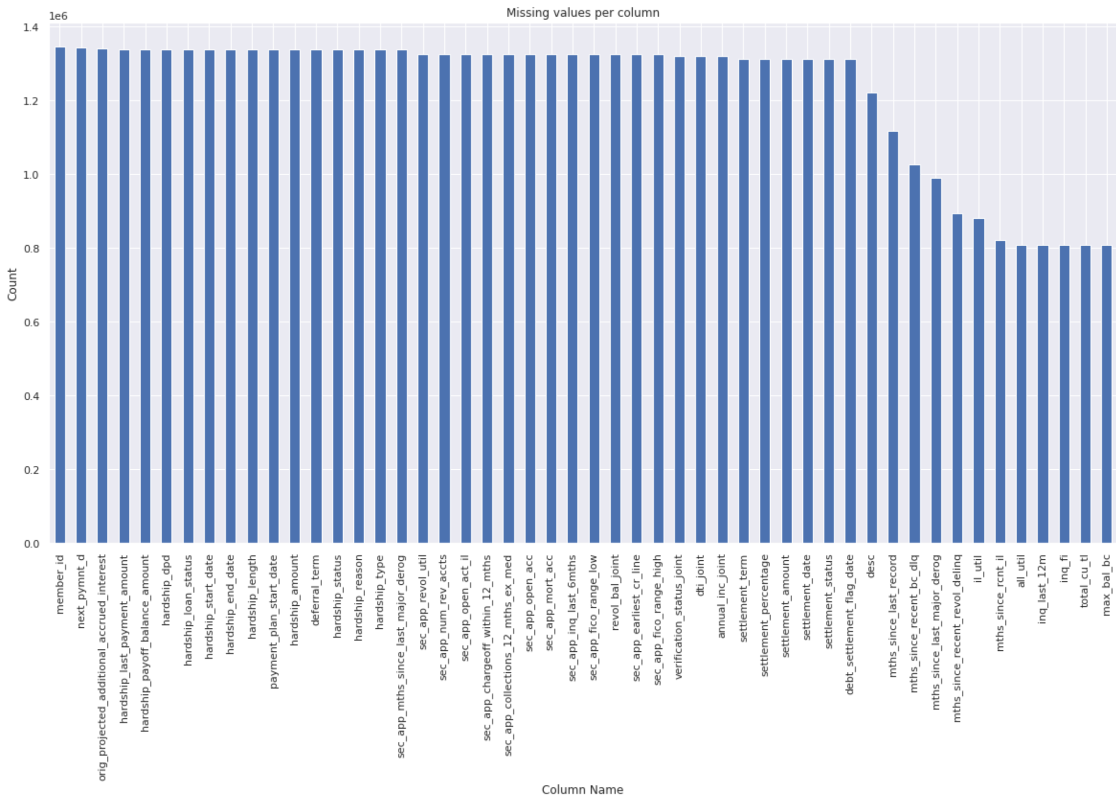

1e6
1.0
0.8
0.6
0.4
0.2
0.0
count
Fully Paid
Charged Off
loan_status


Loan Status
Fully Paid
Charged Off
Fully Paid
80.0%
count
20.0%
Charged Off

1.2 Nulls Distribtion

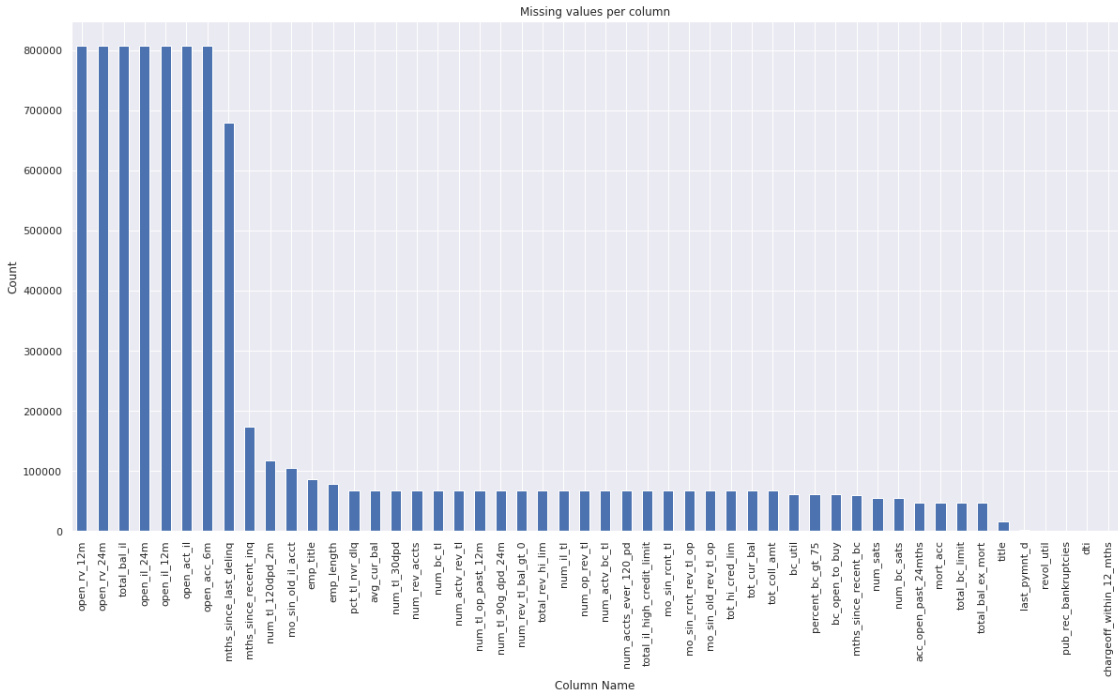
- Get columns which has the most number of null values and sort them

  Visualize the most 50 columns with null values



  The first most 50 columns with highest numbers> of nulls values (Almost all the values are null as number of rows are 1345309 initially) So we have to drop them all as deletion of the rows equivelant to the deletion of most of the data and I can't replace it with any value as most of the values are null, and also if these columns are important they would be filled

  Visualize the next most 50 columns in null values count



The first 95 columns has lots of nulls so I will drop them

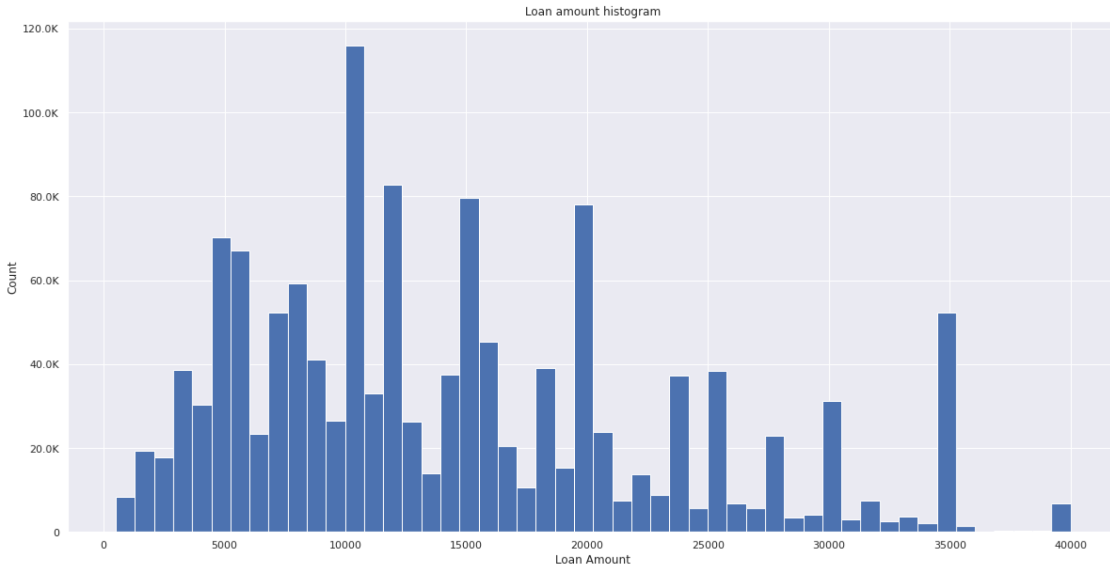
Next, I will drop the rows which has null values they will have a small number of rows

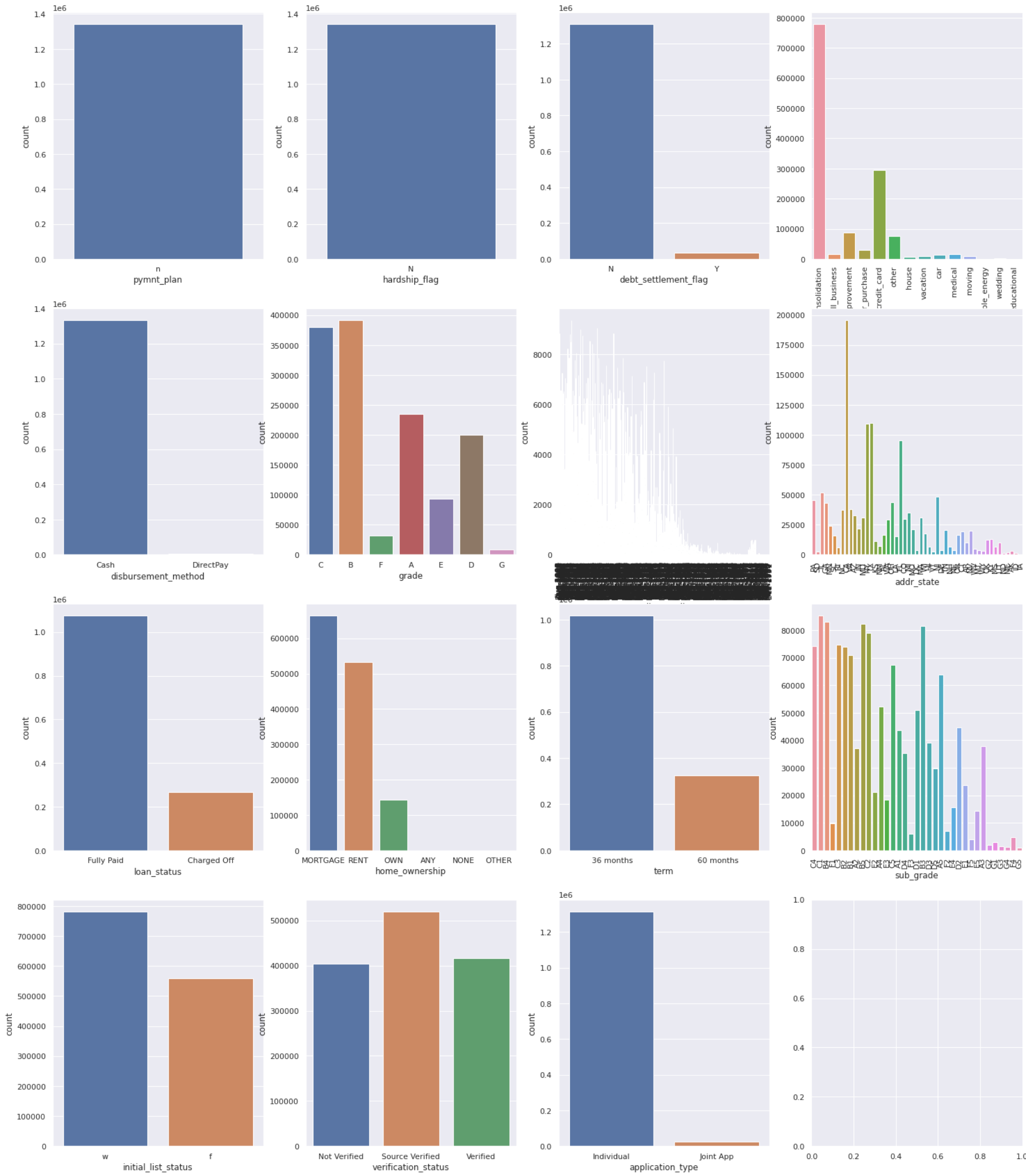
Number of Rows and Columns Now

```
(1340812, 56)
```

Rows Dropped Successfully

1.3 Loan Amount Distribution

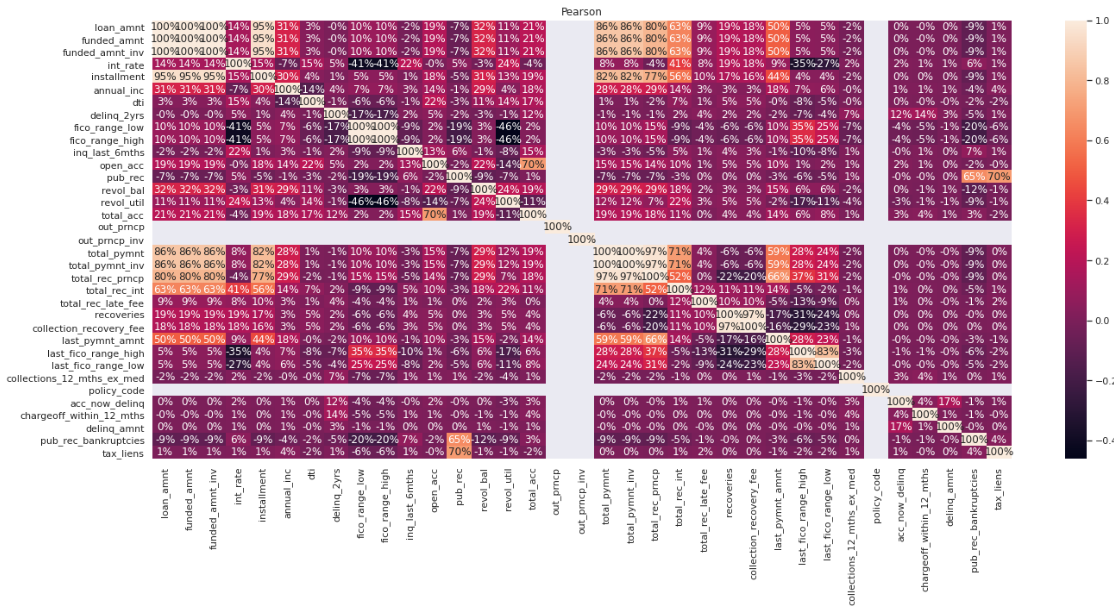




pymnt_plan
hardship_flag
debt_settlement_flag
disbursement_method
grade
addr_state
loan_status
home_ownership
term
sub_grade
initial_list_status
verification_status
application_type
count
1e6
Fully Paid
Charged Off
MORTGAGE
RENT
OWN
ANY
NONE
OTHER
36 months
60 months
Cash
DirectPay
Not Verified
Source Verified
Verified
Individual
Joint App

Drop the followin columns as they are constant columns and doesn't contribute to our prediction of loan_status

Bivariant Visualization



From the previous Correlation Matrix (policy_code, out_prncp, out_prncp_inv) don't have any correlation with any other columns so drop them

Next, Check for redundant information columns: check pairs of features which has correlation value above 0.8

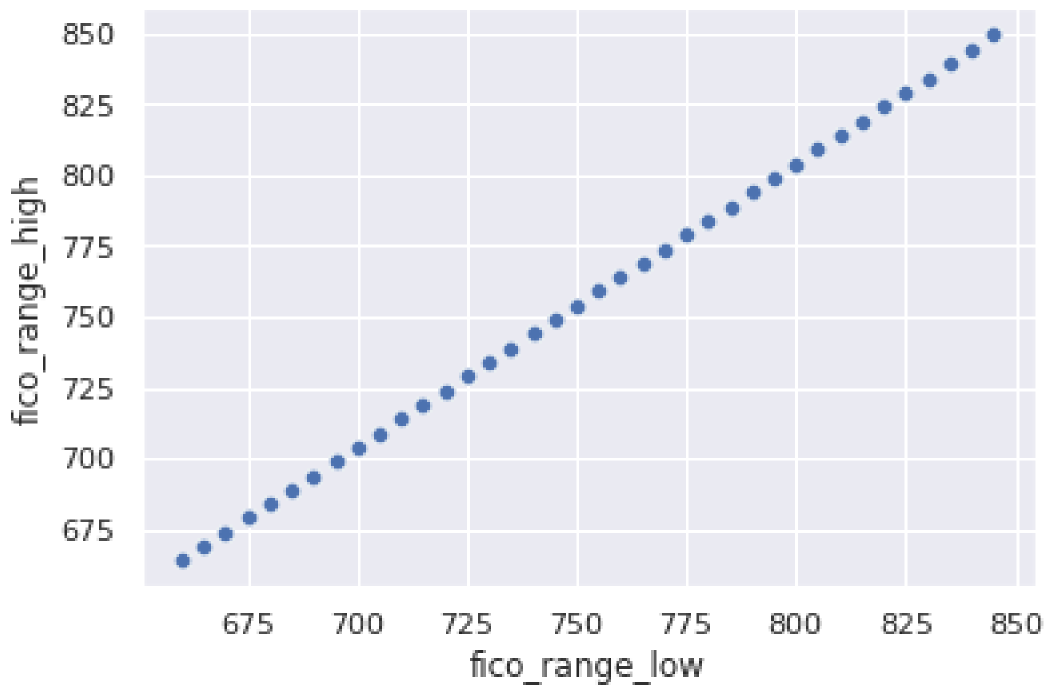
[42]:

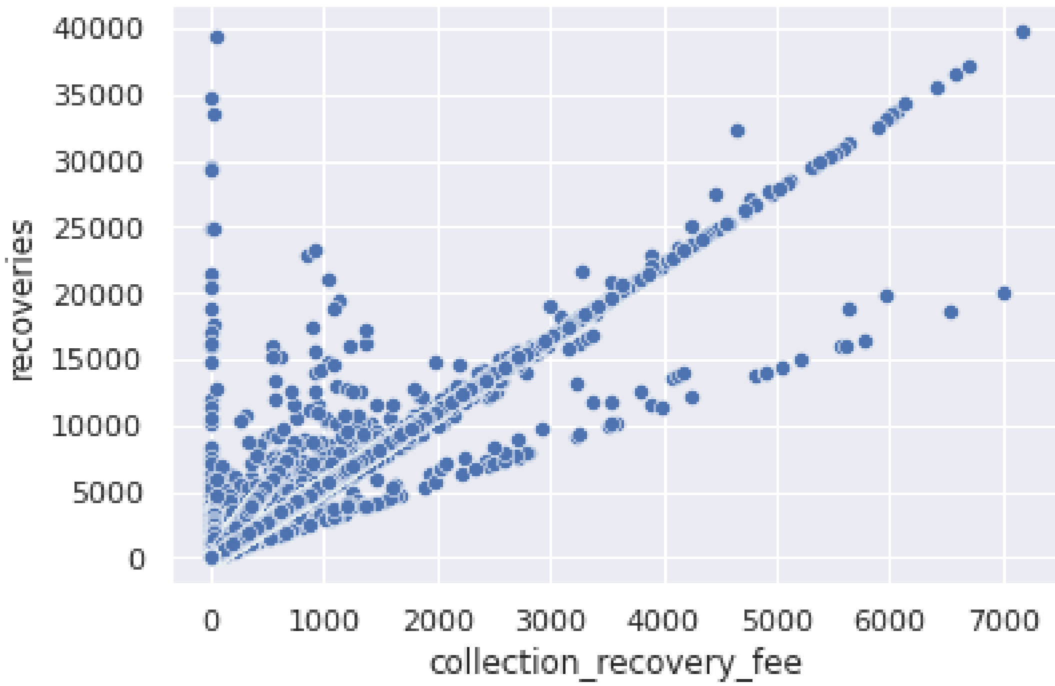
	feature1	feature2	corr
1	fico_range_high	fico_range_low	1.000000
2	loan_amnt	funded_amnt	0.999567
3	total_pymnt_inv	total_pymnt	0.999548
4	funded_amnt_inv	funded_amnt	0.999447
5	funded_amnt_inv	loan_amnt	0.998929
6	collection_recovery_fee	recoveries	0.972815
7	total_rec_prncp	total_pymnt	0.967105
8	total_rec_prncp	total_pymnt_inv	0.966732
9	funded_amnt	installment	0.954036
10	installment	funded_amnt_inv	0.953455
11	installment	loan_amnt	0.953388
12	total_pymnt_inv	funded_amnt_inv	0.857143
13	total_pymnt	funded_amnt	0.856896
14	funded_amnt	total_pymnt_inv	0.856674
15	total_pymnt	loan_amnt	0.856653
16	total_pymnt	funded_amnt_inv	0.856447
17	loan_amnt	total_pymnt_inv	0.856354
18	last_fico_range_low	last_fico_range_high	0.829738
19	installment	total_pymnt	0.818284
20	total_pymnt_inv	installment	0.818049

It seems that the data have many duplicated information represented by multiple columns, so let's drop these duplicated columns

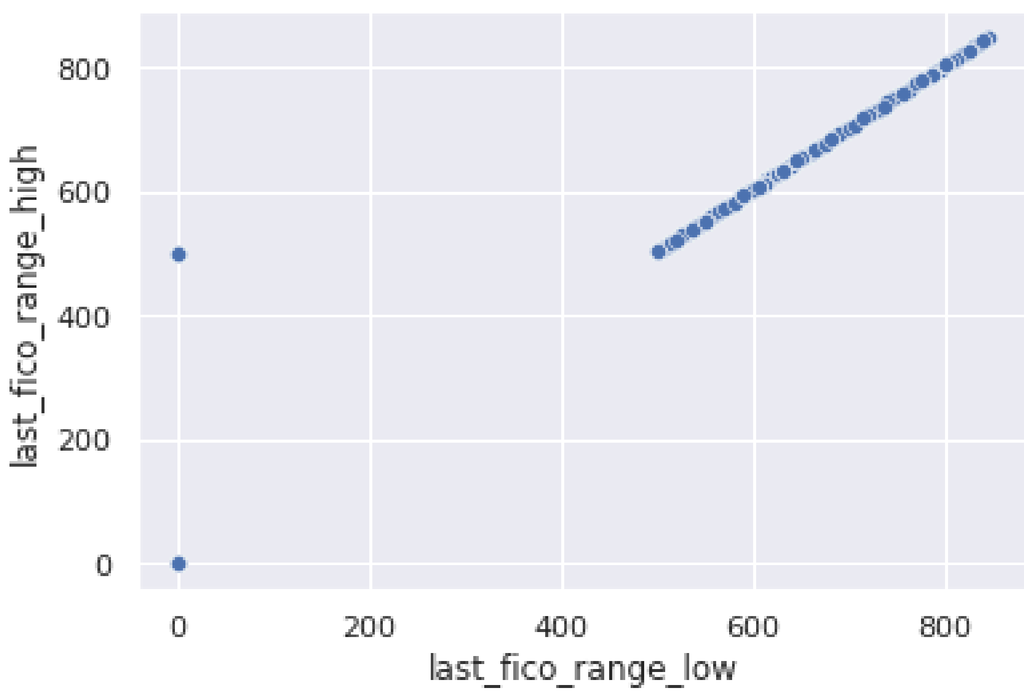
- fico_range_low = (fico_range_high)
- funded_amnt = funded_amnt_inv = installment = total_pymnt_inv = total_rec_prncp = total_pymnt = (loan_amnt)
- collection_recovery_fee = (recoveries)
- last_fico_range_low = (last_fico_range_high)

  Drop them all except the columns between brackets () to avoid information redundancy

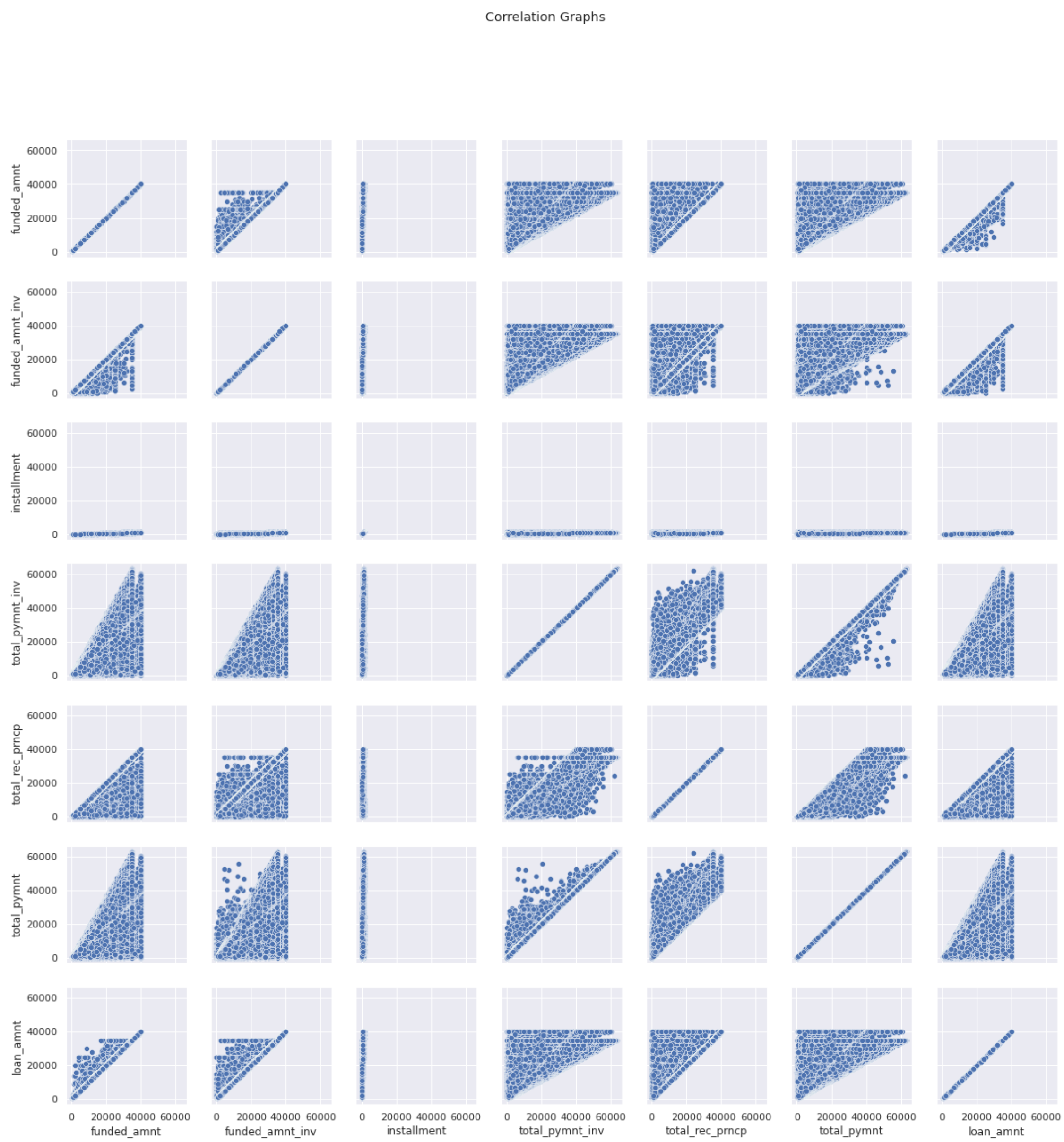




40000
35000
30000
25000
20000
15000
10000
5000
0
recoveries
0
1000
2000
3000
4000
5000
6000
7000
collection_recovery_fee



800
600
400
200
0
last_fico_range_high
0
200
400
600
800
last_fico_range_low



After dropping these columns

```
(1340812, 41)
```

Preprocessing

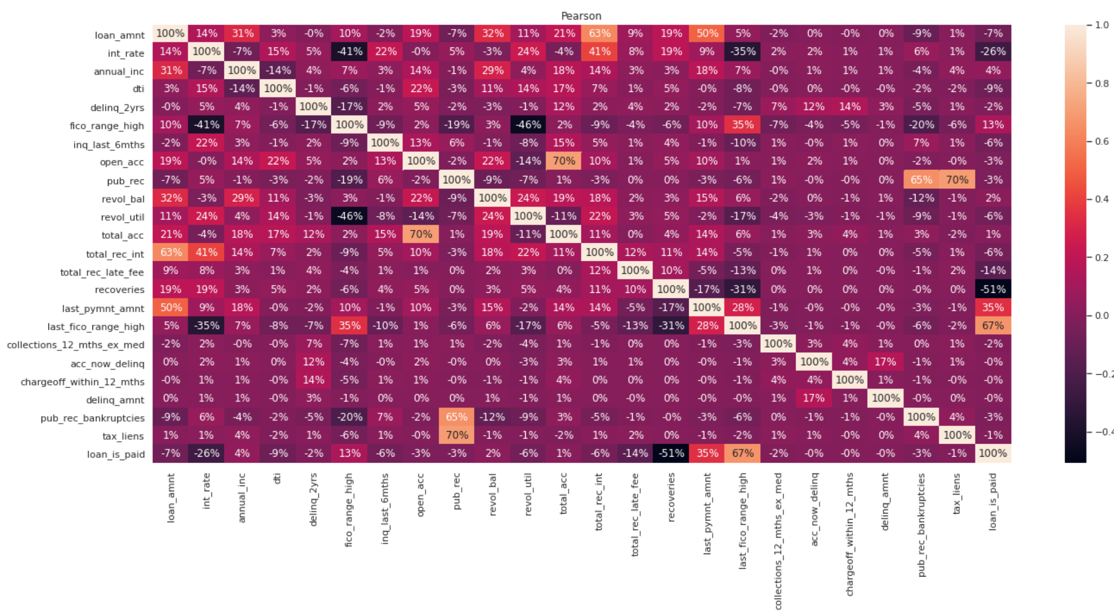
1.3.1 loan_status Column

Map loan_status to new column called loan_is_paid has 1 and 0 values only, 1 for Fully Paid and 0 for Charged Off, and then drop the old loan_status column

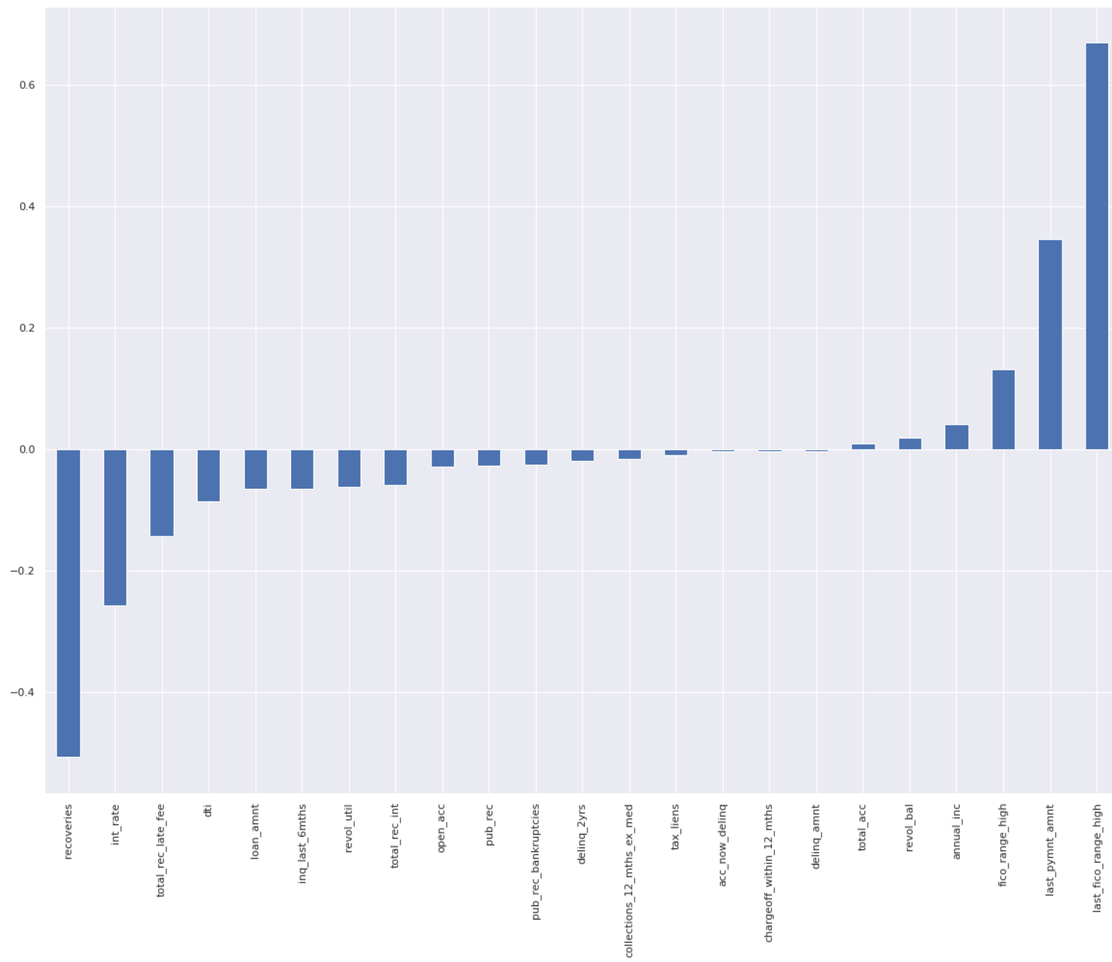
```
+------------+-------+
|loan_is_paid|  count|
```

```
+------------+-------+
|           1|1074961|
|           0| 265851|
+------------+-------+
```

New Correlation matrix



Correlation between loan_is_paid and all other numeric columns



1.3.2 term Column

map term column to new term_months columns with mapped values from ' 36 months' to 36 and from ' 60 months' to 60, and then drop the old term column

[44]:
```
[' 36 months', ' 60 months']
```

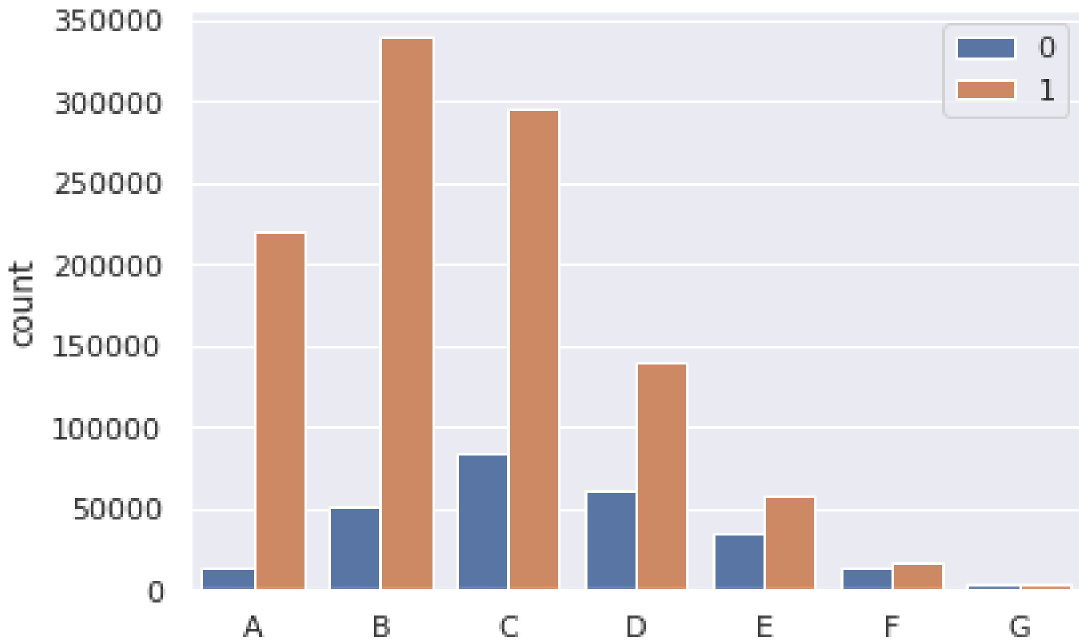
```
+-----------+
|term_months|
+-----------+
|         60|
|         36|
+-----------+
```

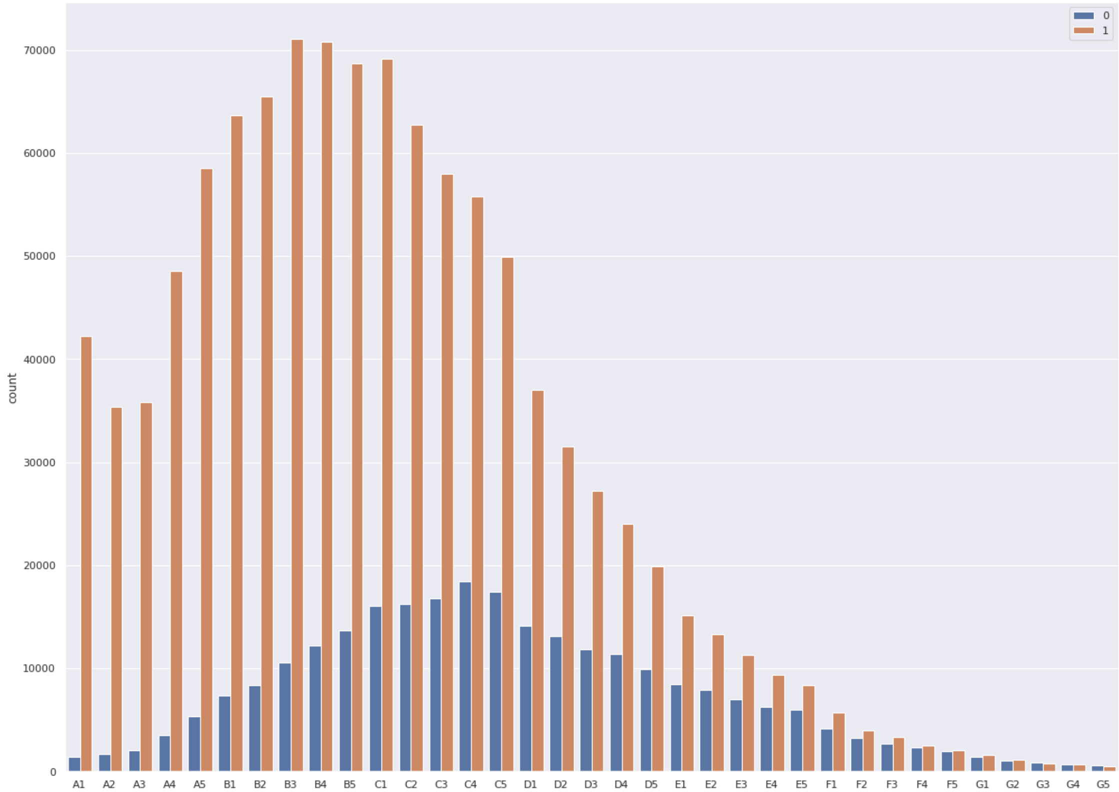
1.3.3 home_ownership Column

We can merge NONE with ANY in one category

```
+--------------+------+
|home_ownership| count|
+--------------+------+
|           OWN|144179|
|          RENT|532381|
|      MORTGAGE|663782|
|           ANY|   328|
|         OTHER|   142|
+--------------+------+
```

1.3.4 grade And sub_grade Columns





grade is part of sub_grade, so let's drop it

Date columns: issue_d, last_pymnt_d, last_credit_pull_d are not important to the analysis

earliest_cr_line which is the month when reported credit line was opened is not important to the analysis

url for LC page with listing data is not important to the analysis

addresses: zip_code, addr_state are not important to the analysis

1.4 Handle Categorical Features

1.4.1 Spark Pipeline

1. Categorical columns to string indexer to change categories to numbers
2. OneHotEncode these new numbers to (#num of column - 1) new column every column value with has 1 whenever this category happens to be in this row
3. Assemble all the features the onehotencoded and the numeric columns in one vector columns used as feature column
4. Get scaled_feature column by scaling feature column using MinMaxcaler

Select now the two important columns used in building the model: - scaled_features - loan_is_paid

```
root
 |-- scaled_features: vector (nullable = true)
 |-- loan_is_paid: integer (nullable = true)


+--------------------+------------+
|     scaled_features|loan_is_paid|
+--------------------+------------+
|(81,[1,2,4,5,6,8,...|           1|
+--------------------+------------+
only showing top 1 row
```

From previous table the number of scaled_features are 81 feature

Deeplearning Model

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 78)                6396
_________________________________________________________________
dense_7 (Dense)              (None, 39)                3081
_________________________________________________________________
dense_8 (Dense)              (None, 19)                760
_________________________________________________________________
dense_9 (Dense)              (None, 8)                 160
_________________________________________________________________
dense_10 (Dense)             (None, 4)                 36
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 5
=================================================================
Total params: 10,438
Trainable params: 10,438
Non-trainable params: 0
_________________________________________________________________
```

Transform to pandas dataframe before training

Split data 80% training set and 20% testing set

Train the model

```
Epoch 1/40
2096/2096 [==============================] - 33s 10ms/step - loss: 0.5434 -
accuracy: 0.8413 - val_loss: 0.1932 - val_accuracy: 0.9663
Epoch 2/40
2096/2096 [==============================] - 23s 11ms/step - loss: 0.1654 -
accuracy: 0.9668 - val_loss: 0.1250 - val_accuracy: 0.9628
Epoch 3/40
2096/2096 [==============================] - 25s 12ms/step - loss: 0.1009 -
```

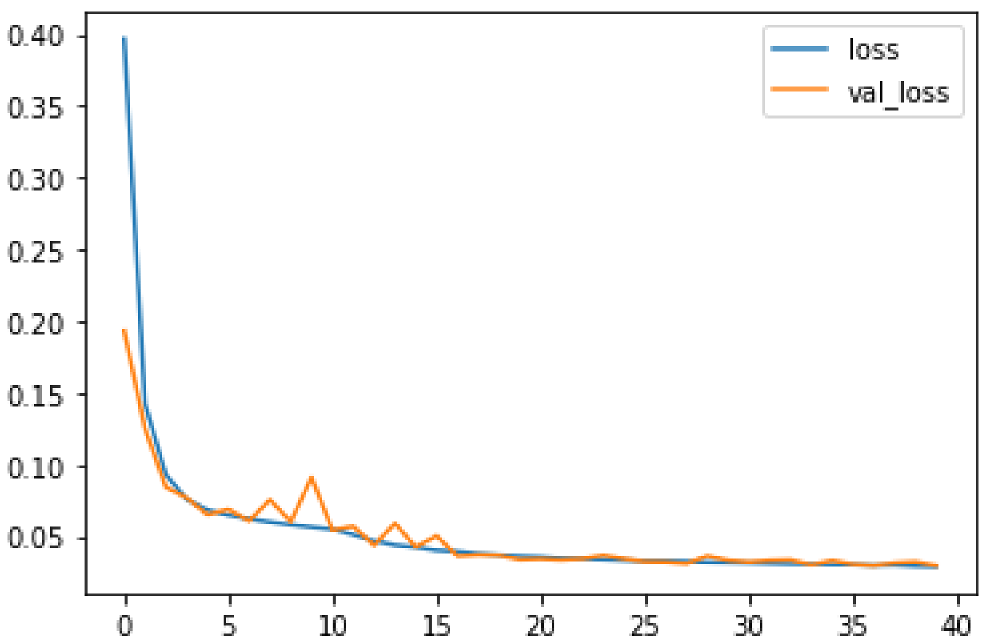
```
accuracy: 0.9713 - val_loss: 0.0843 - val_accuracy: 0.9722
Epoch 4/40
2096/2096 [==============================] - 27s 13ms/step - loss: 0.0783 -
accuracy: 0.9740 - val_loss: 0.0769 - val_accuracy: 0.9717
Epoch 5/40
2096/2096 [==============================] - 28s 13ms/step - loss: 0.0690 -
accuracy: 0.9757 - val_loss: 0.0650 - val_accuracy: 0.9766
Epoch 6/40
2096/2096 [==============================] - 29s 14ms/step - loss: 0.0650 -
accuracy: 0.9765 - val_loss: 0.0688 - val_accuracy: 0.9737
Epoch 7/40
2096/2096 [==============================] - 31s 15ms/step - loss: 0.0623 -
accuracy: 0.9772 - val_loss: 0.0607 - val_accuracy: 0.9778
Epoch 8/40
2096/2096 [==============================] - 29s 14ms/step - loss: 0.0607 -
accuracy: 0.9779 - val_loss: 0.0755 - val_accuracy: 0.9728
Epoch 9/40
2096/2096 [==============================] - 33s 16ms/step - loss: 0.0588 -
accuracy: 0.9786 - val_loss: 0.0601 - val_accuracy: 0.9786
Epoch 10/40
2096/2096 [==============================] - 33s 16ms/step - loss: 0.0572 -
accuracy: 0.9794 - val_loss: 0.0912 - val_accuracy: 0.9629
Epoch 11/40
2096/2096 [==============================] - 34s 16ms/step - loss: 0.0572 -
accuracy: 0.9792 - val_loss: 0.0543 - val_accuracy: 0.9808
Epoch 12/40
2096/2096 [==============================] - 30s 15ms/step - loss: 0.0518 -
accuracy: 0.9815 - val_loss: 0.0569 - val_accuracy: 0.9799
Epoch 13/40
2096/2096 [==============================] - 37s 18ms/step - loss: 0.0485 -
accuracy: 0.9825 - val_loss: 0.0437 - val_accuracy: 0.9843
Epoch 14/40
2096/2096 [==============================] - 28s 13ms/step - loss: 0.0444 -
accuracy: 0.9843 - val_loss: 0.0590 - val_accuracy: 0.9803
Epoch 15/40
2096/2096 [==============================] - 28s 13ms/step - loss: 0.0449 -
accuracy: 0.9841 - val_loss: 0.0425 - val_accuracy: 0.9848
Epoch 16/40
2096/2096 [==============================] - 32s 15ms/step - loss: 0.0404 -
accuracy: 0.9856 - val_loss: 0.0503 - val_accuracy: 0.9824
Epoch 17/40
2096/2096 [==============================] - 35s 17ms/step - loss: 0.0410 -
accuracy: 0.9855 - val_loss: 0.0362 - val_accuracy: 0.9871
Epoch 18/40
2096/2096 [==============================] - 33s 16ms/step - loss: 0.0374 -
accuracy: 0.9866 - val_loss: 0.0370 - val_accuracy: 0.9867
Epoch 19/40
2096/2096 [==============================] - 34s 16ms/step - loss: 0.0371 -
```

```
accuracy: 0.9867 - val_loss: 0.0367 - val_accuracy: 0.9868
Epoch 20/40
2096/2096 [==============================] - 34s 16ms/step - loss: 0.0364 -
accuracy: 0.9869 - val_loss: 0.0337 - val_accuracy: 0.9877
Epoch 21/40
2096/2096 [==============================] - 35s 17ms/step - loss: 0.0355 -
accuracy: 0.9872 - val_loss: 0.0342 - val_accuracy: 0.9878
Epoch 22/40
2096/2096 [==============================] - 34s 16ms/step - loss: 0.0346 -
accuracy: 0.9876 - val_loss: 0.0335 - val_accuracy: 0.9880
Epoch 23/40
2096/2096 [==============================] - 37s 18ms/step - loss: 0.0343 -
accuracy: 0.9877 - val_loss: 0.0343 - val_accuracy: 0.9875
Epoch 24/40
2096/2096 [==============================] - 34s 16ms/step - loss: 0.0333 -
accuracy: 0.9881 - val_loss: 0.0367 - val_accuracy: 0.9866
Epoch 25/40
2096/2096 [==============================] - 36s 17ms/step - loss: 0.0335 -
accuracy: 0.9880 - val_loss: 0.0343 - val_accuracy: 0.9875
Epoch 26/40
2096/2096 [==============================] - 35s 17ms/step - loss: 0.0329 -
accuracy: 0.9883 - val_loss: 0.0323 - val_accuracy: 0.9884
Epoch 27/40
2096/2096 [==============================] - 38s 18ms/step - loss: 0.0326 -
accuracy: 0.9885 - val_loss: 0.0320 - val_accuracy: 0.9885
Epoch 28/40
2096/2096 [==============================] - 38s 18ms/step - loss: 0.0320 -
accuracy: 0.9886 - val_loss: 0.0309 - val_accuracy: 0.9890
Epoch 29/40
2096/2096 [==============================] - 37s 18ms/step - loss: 0.0320 -
accuracy: 0.9885 - val_loss: 0.0363 - val_accuracy: 0.9866
Epoch 30/40
2096/2096 [==============================] - 35s 17ms/step - loss: 0.0315 -
accuracy: 0.9887 - val_loss: 0.0332 - val_accuracy: 0.9880
Epoch 31/40
2096/2096 [==============================] - 39s 18ms/step - loss: 0.0310 -
accuracy: 0.9888 - val_loss: 0.0320 - val_accuracy: 0.9884
Epoch 32/40
2096/2096 [==============================] - 39s 19ms/step - loss: 0.0311 -
accuracy: 0.9889 - val_loss: 0.0332 - val_accuracy: 0.9881
Epoch 33/40
2096/2096 [==============================] - 35s 17ms/step - loss: 0.0300 -
accuracy: 0.9893 - val_loss: 0.0335 - val_accuracy: 0.9879
Epoch 34/40
2096/2096 [==============================] - 38s 18ms/step - loss: 0.0307 -
accuracy: 0.9889 - val_loss: 0.0301 - val_accuracy: 0.9892
Epoch 35/40
2096/2096 [==============================] - 39s 19ms/step - loss: 0.0304 -
```

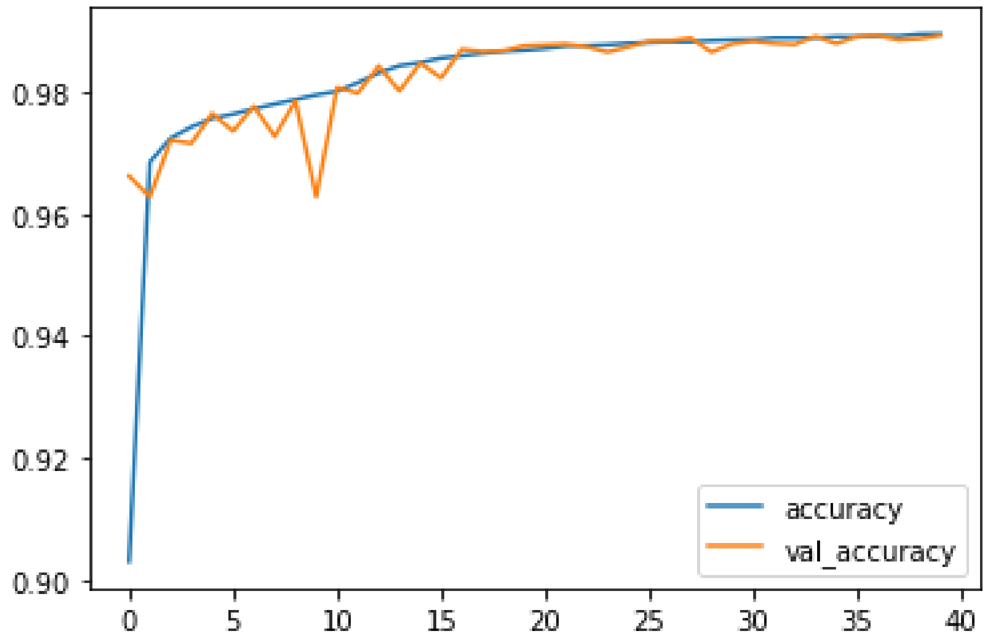
```
accuracy: 0.9890 - val_loss: 0.0330 - val_accuracy: 0.9880
Epoch 36/40
2096/2096 [==============================] - 33s 16ms/step - loss: 0.0295 -
accuracy: 0.9894 - val_loss: 0.0304 - val_accuracy: 0.9892
Epoch 37/40
2096/2096 [==============================] - 43s 21ms/step - loss: 0.0292 -
accuracy: 0.9895 - val_loss: 0.0296 - val_accuracy: 0.9894
Epoch 38/40
2096/2096 [==============================] - 41s 20ms/step - loss: 0.0293 -
accuracy: 0.9894 - val_loss: 0.0314 - val_accuracy: 0.9886
Epoch 39/40
2096/2096 [==============================] - 42s 20ms/step - loss: 0.0289 -
accuracy: 0.9898 - val_loss: 0.0321 - val_accuracy: 0.9888
Epoch 40/40
2096/2096 [==============================] - 41s 20ms/step - loss: 0.0292 -
accuracy: 0.9895 - val_loss: 0.0295 - val_accuracy: 0.9894
```

[69]: <tensorflow.python.keras.callbacks.History at 0x7fedb1dde050>

[71]: <AxesSubplot:>



[72]: <AxesSubplot:>



Accuracy = 98.94%

Saving the model ...

```
INFO:tensorflow:Assets written to: loan_prediction_model/assets
```

Gradient Boosting Tree (spark)

Using the map reduce machine learning models of pyspark we can build model using HDFS

Split data 80% training set and 20% testing set

Saving the model ...

Some Predictions

```
+--------------------+------------+----------+--------------------+
|     scaled_features|loan_is_paid|prediction|         probability|
+--------------------+------------+----------+--------------------+
|(81,[0,1,2,3,4,5,...|           1|       1.0|[0.05270399993585...|
|(81,[0,1,2,3,4,5,...|           1|       1.0|[0.05443185145658...|
|(81,[0,1,2,3,4,5,...|           0|       0.0|[0.95635347857271...|
|(81,[0,1,2,3,4,5,...|           0|       0.0|[0.95635347857271...|
|(81,[0,1,2,3,4,5,...|           0|       0.0|[0.95635347857271...|
|(81,[0,1,2,3,4,5,...|           0|       0.0|[0.95635347857271...|
|(81,[0,1,2,3,4,5,...|           0|       0.0|[0.95635347857271...|
|(81,[0,1,2,3,4,5,...|           1|       1.0|[0.04368680010337...|
|(81,[0,1,2,3,4,5,...|           1|       1.0|[0.08128007245037...|
|(81,[0,1,2,3,4,5,...|           0|       1.0|[0.27032648846438...|
```

```
+--------------------+------------+----------+--------------------+
only showing top 10 rows

Test Area Under ROC: 0.9524807005159022

Test f1 score:  0.9744686686161473

Test accuracy:  0.9746691093995363

+------------+----------+------+
|loan_is_paid|prediction| count|
+------------+----------+------+
|           1|       0.0|  2304|
|           0|       0.0| 48570|
|           1|       1.0|212115|
|           0|       1.0|  4471|
+------------+----------+------+
```

Previous table has FP, FN, TP and TN values