# Bigdata Project Report

June 9, 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.

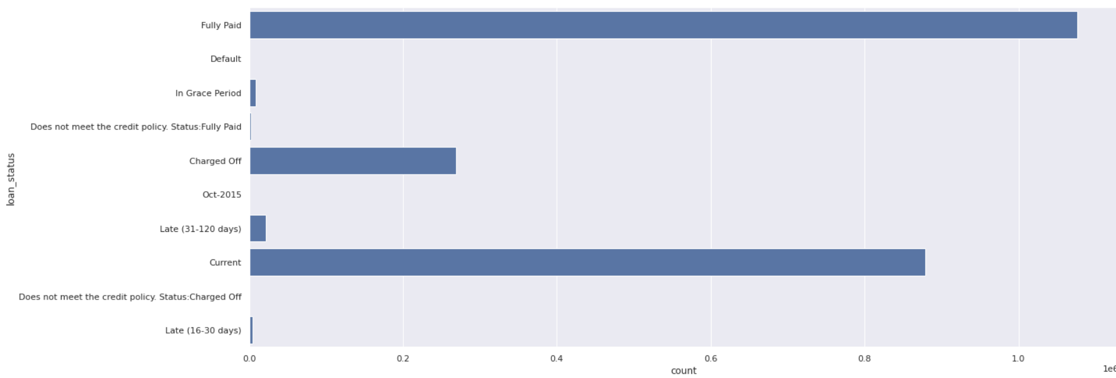
Objectives of this notebook is: - To show step-by-step how to visualize the dataset. - Data cleaning and preprocessing. - Assess whether or not a new customer is likely to pay back the loan.

# Univariant Visualization

### 1.1 Loan Status Distribution

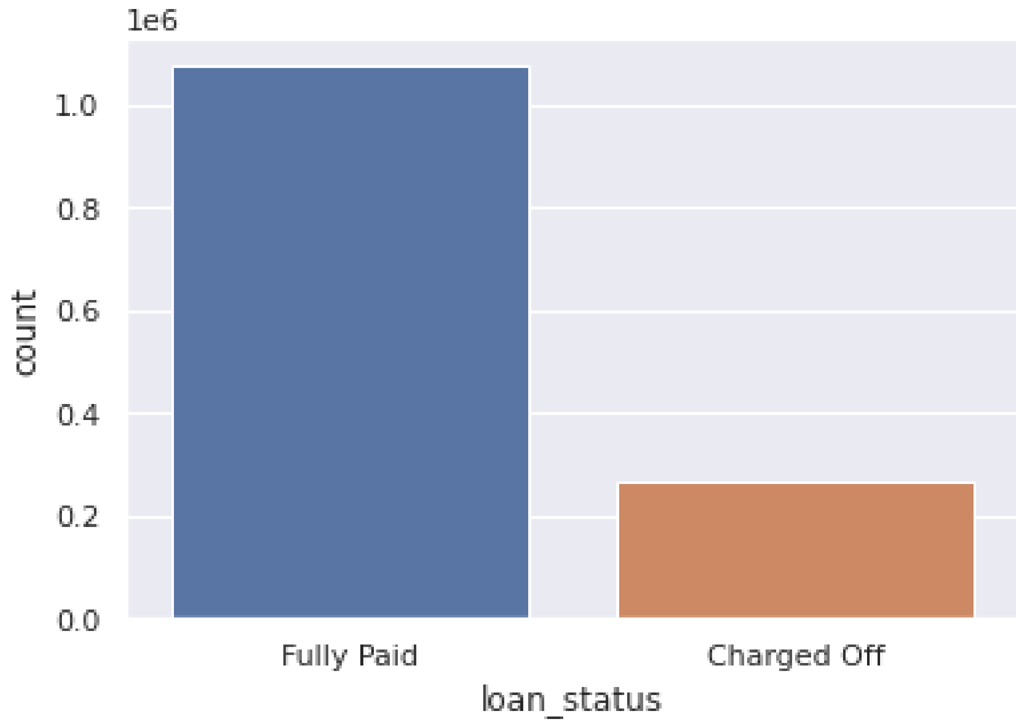
```
+--------------------+-------+
|         loan_status|  count|
+--------------------+-------+
|          Fully Paid|1076751|
|             Default|     40|
|                null|     33|
|     In Grace Period|   8436|
|Does not meet the...|   1988|
```

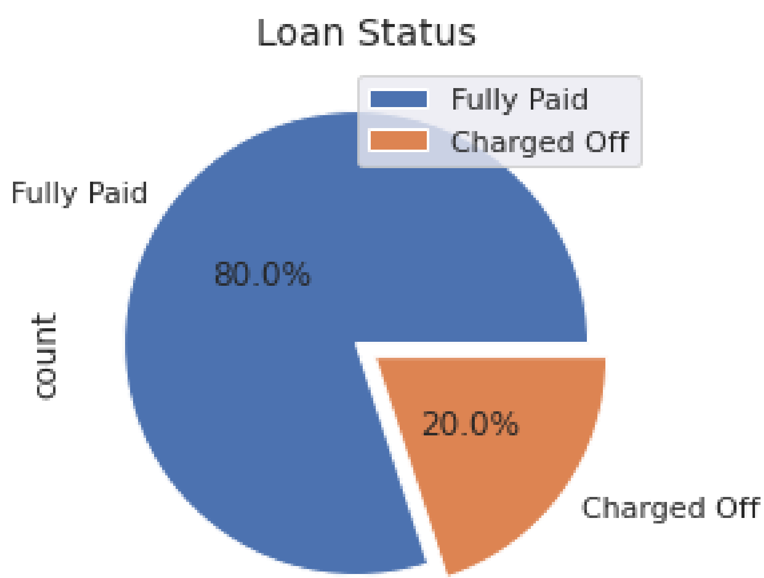
```
|         Charged Off| 268558|
|            Oct-2015|      1|
|  Late (31-120 days)|  21467|
|             Current| 878317|
|Does not meet the...|    761|
|   Late (16-30 days)|   4349|
+--------------------+-------+
```




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

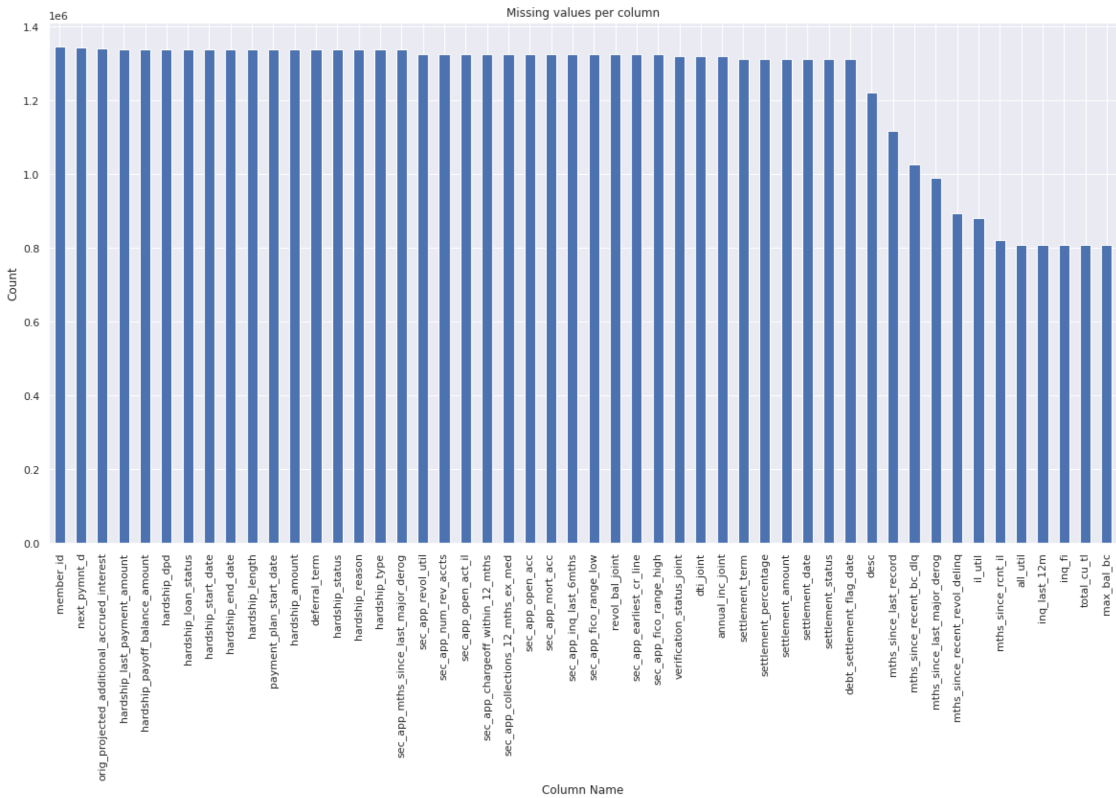
```
+-----------+-------+
|loan_status|  count|
+-----------+-------+
| Fully Paid|1076751|
|Charged Off| 268558|
+-----------+-------+
```


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

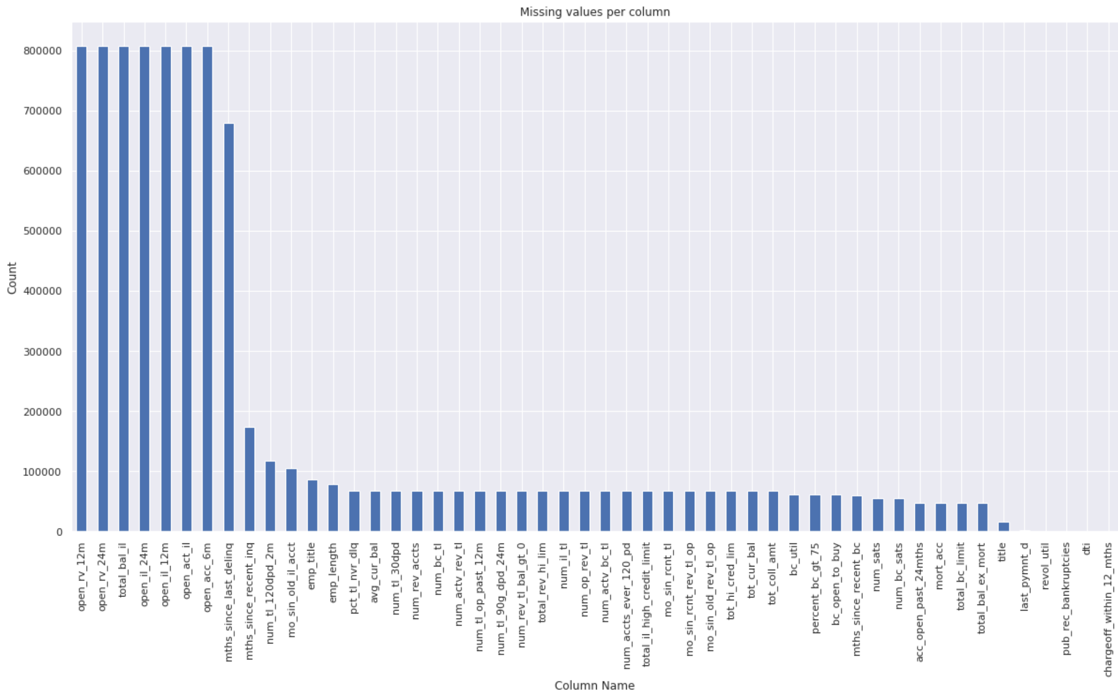
### 1.2 Handling Null Values

- Get columns which has the most number of null values and sort them
- Visualize the most 50 columns with null values



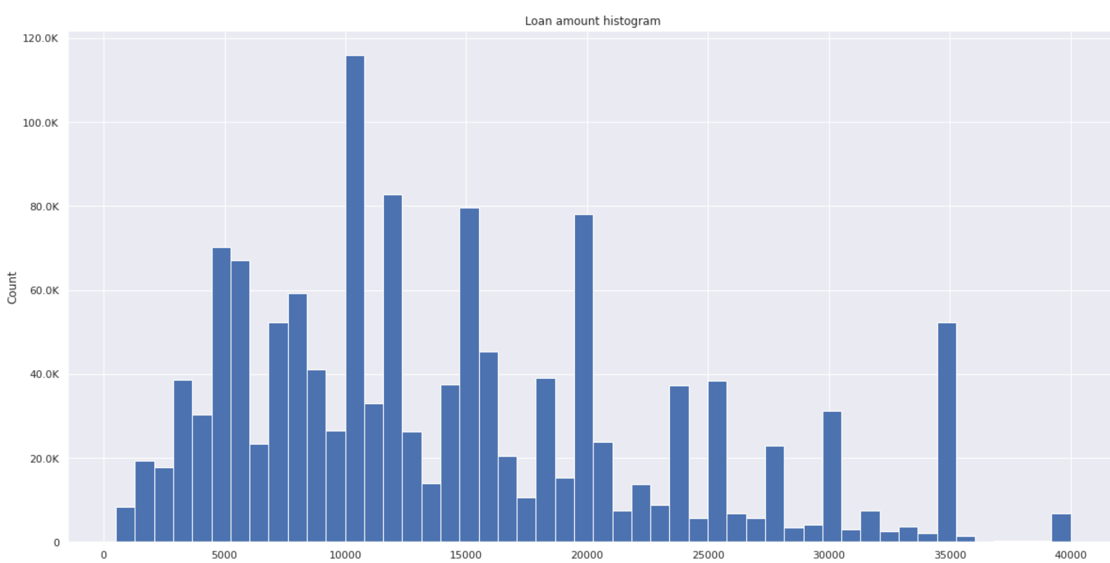

The first most 50 columns with highest numbers of null values (Almost all the values are null as number of rows are 1345309 initially) So we have to drop them all as deletion of the rows means the deletion of most of the data and I can't replace it mean value as most of the values are null it won't be the best option, and also if these columns are important they would be filled, so We will drop them

- Next, We visualize the next most 50 columns have null values count

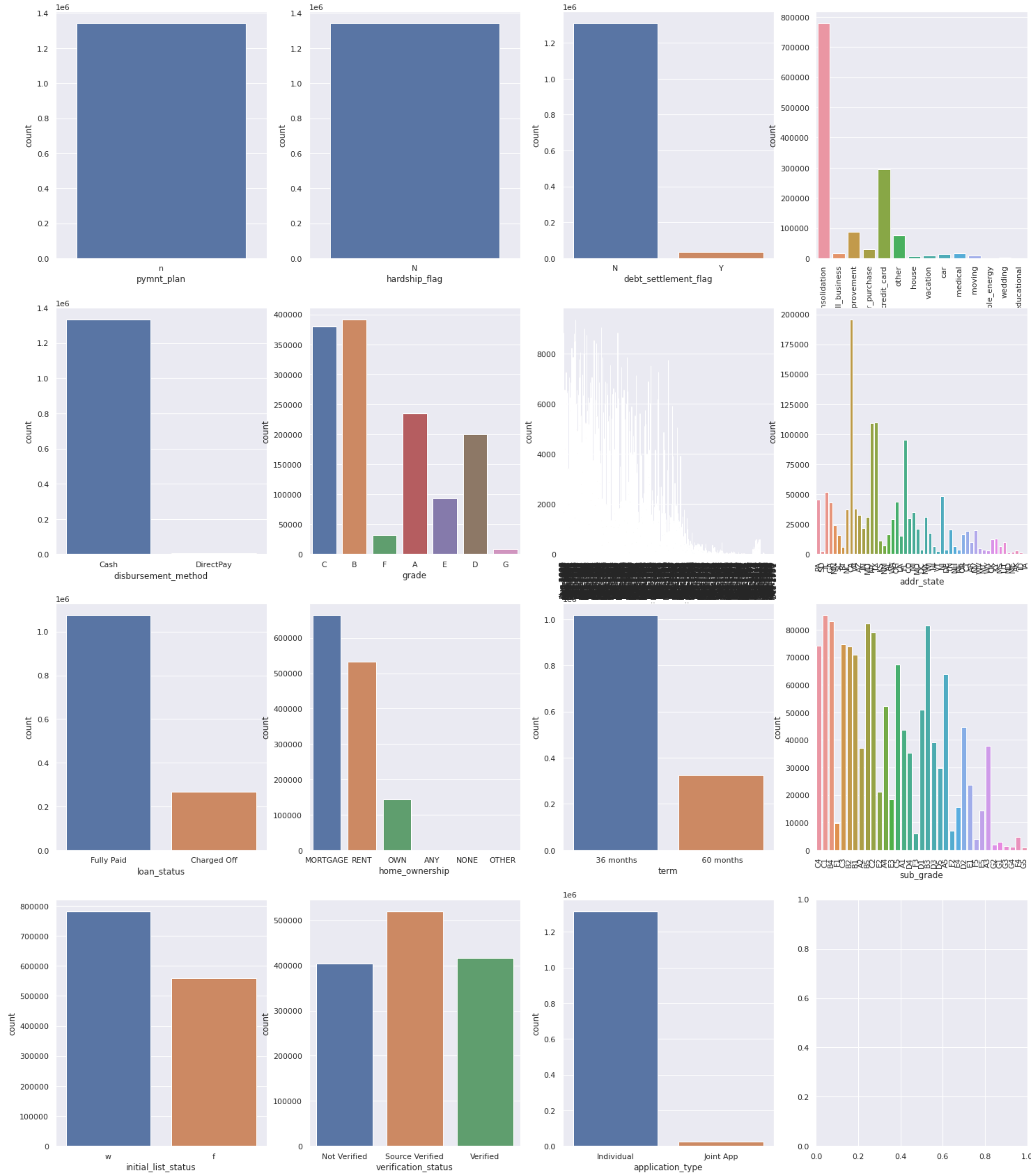



- From the previous graph the first 95 columns has lots of nulls so I will drop them
- Next, We will drop the rows which has null values they will have a small number of rows
- Now the Number Of Columns = 56, Number Of Rows = 1340812

## 1.3 Loan Amount Distribution

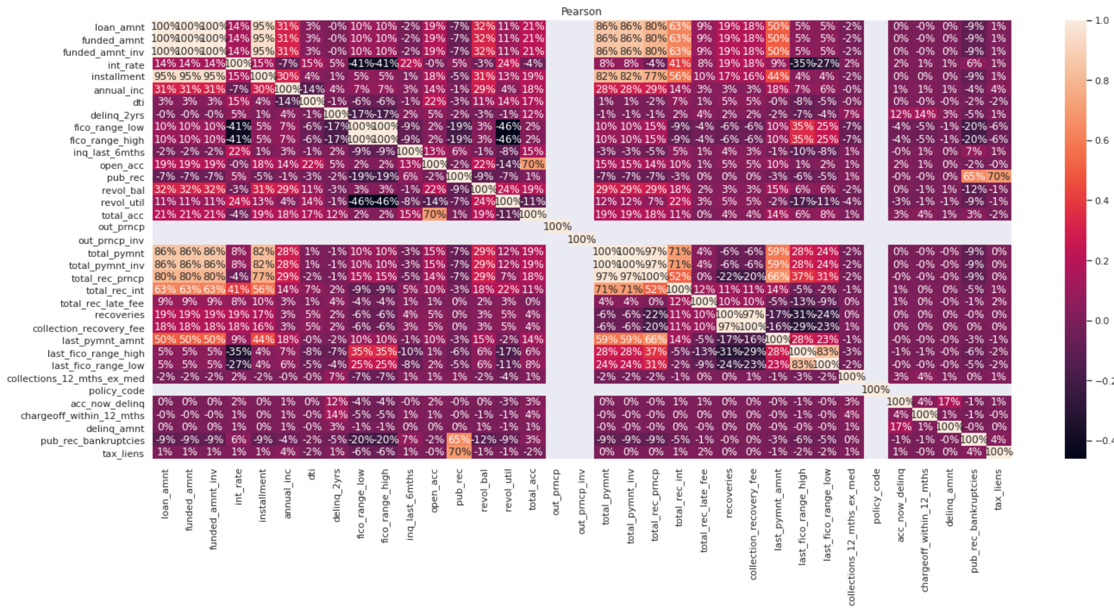


- Most of loan amount 10,000 to 11,000 usd




From the above graphs, We can deduce that we have duplicates information for example, 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

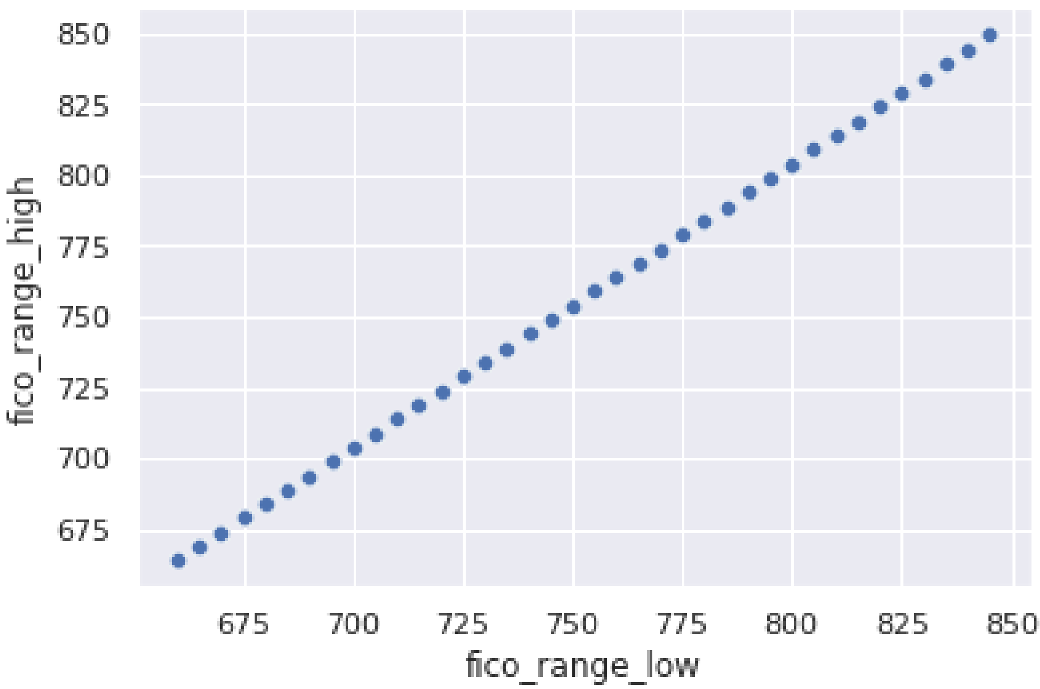
[42]:

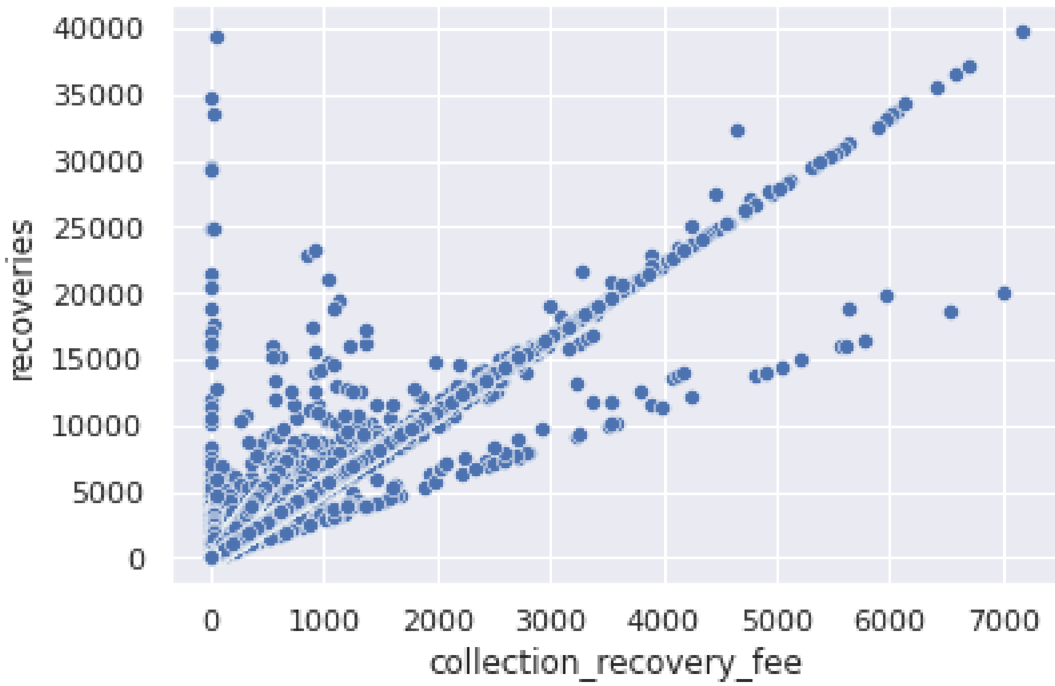
| | feature1 | feature2 | corr |
|---|---|---|---|
| 1 | fico_range_high | fico_range_low | 1.000000 |
| 2 | loan_amnt | funded_amnt | 0.999567 |
| 3 | total_pymnt_inv | total_pymnt | 0.999548 |
| 4 | funded_amnt_inv | funded_amnt | 0.999447 |
| 5 | funded_amnt_inv | loan_amnt | 0.998929 |
| 6 | collection_recovery_fee | 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

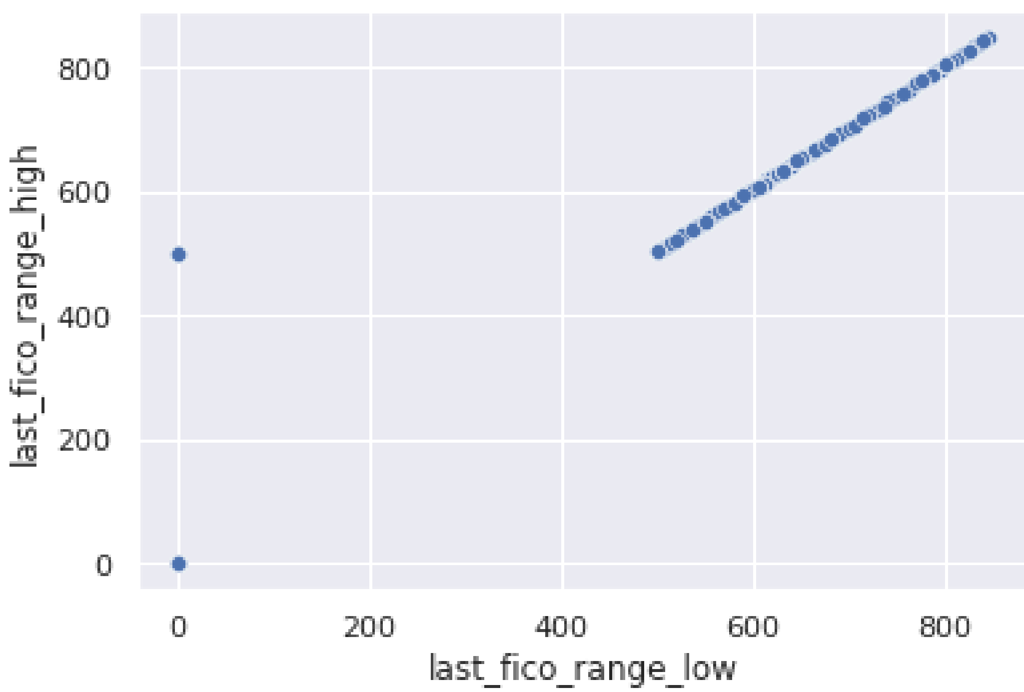
- fico_range_low = (fico_range_high)
- funded_amnt = funded_amnt_inv = installment = total_pymnt_inv = total_rec_prncp = total_pymnt = (loan_amnt)
- collection_recovery_fee = (recoveries)
- last_fico_range_low = (last_fico_range_high)

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



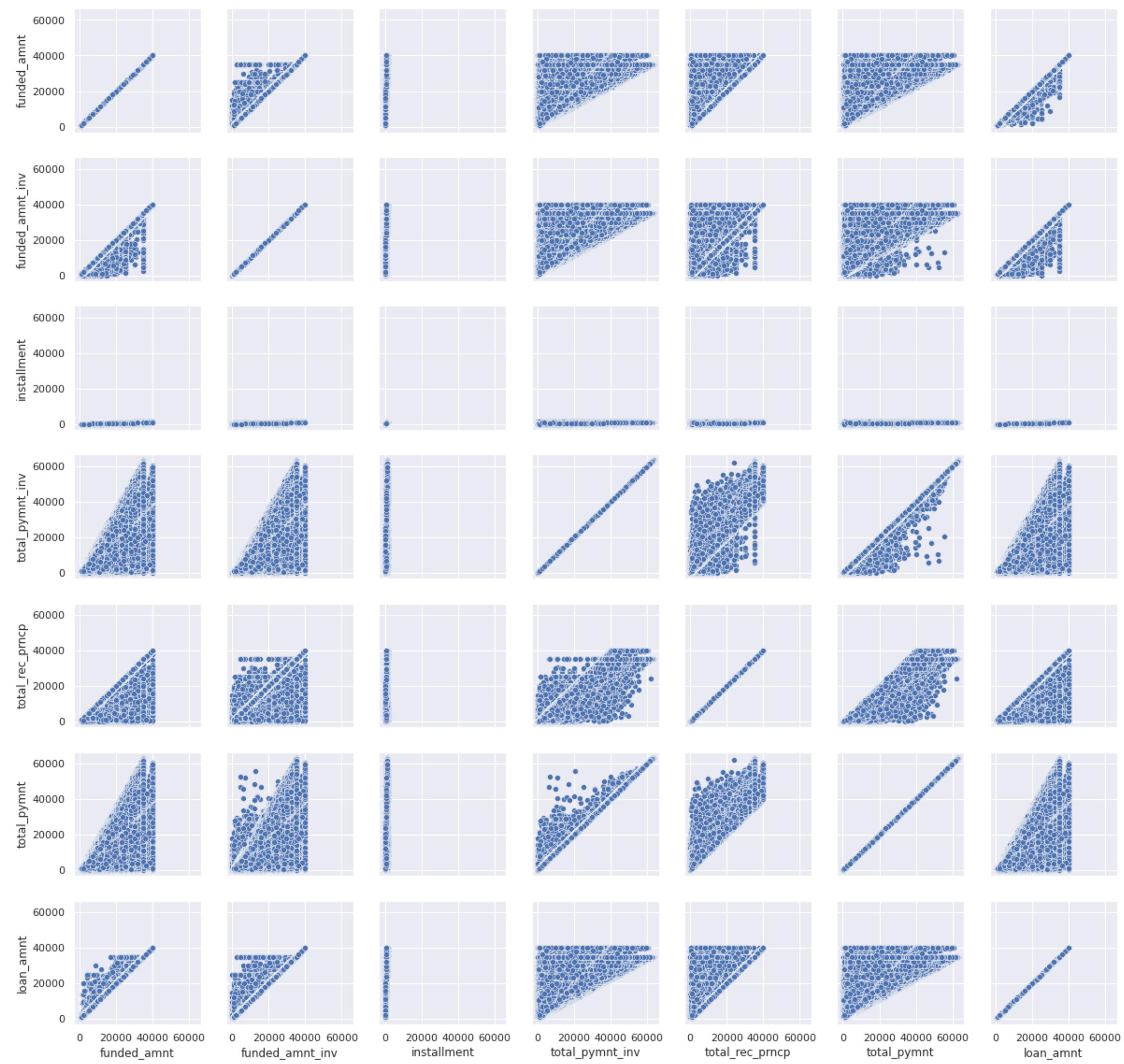


40000
35000
30000
25000
20000
15000
10000
5000
0
recoveries
0
1000
2000
3000
4000
5000
6000
7000
collection_recovery_fee

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

Correlation Graphs




After dropping these columns

```
(1340812, 41)
```

# Preprocessing

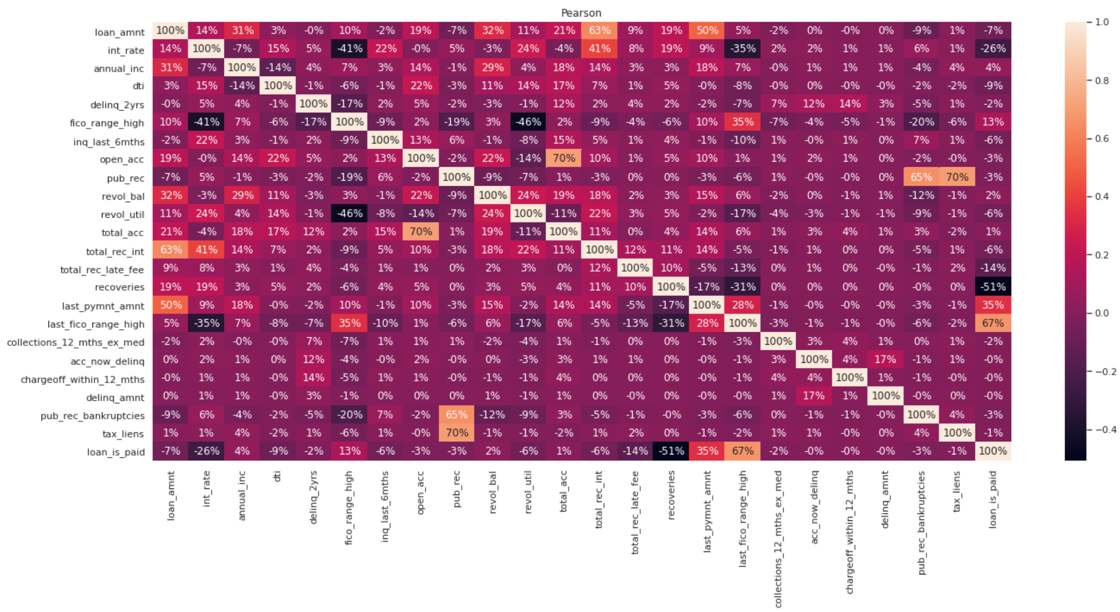
### 1.3.1 loan_status Column

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

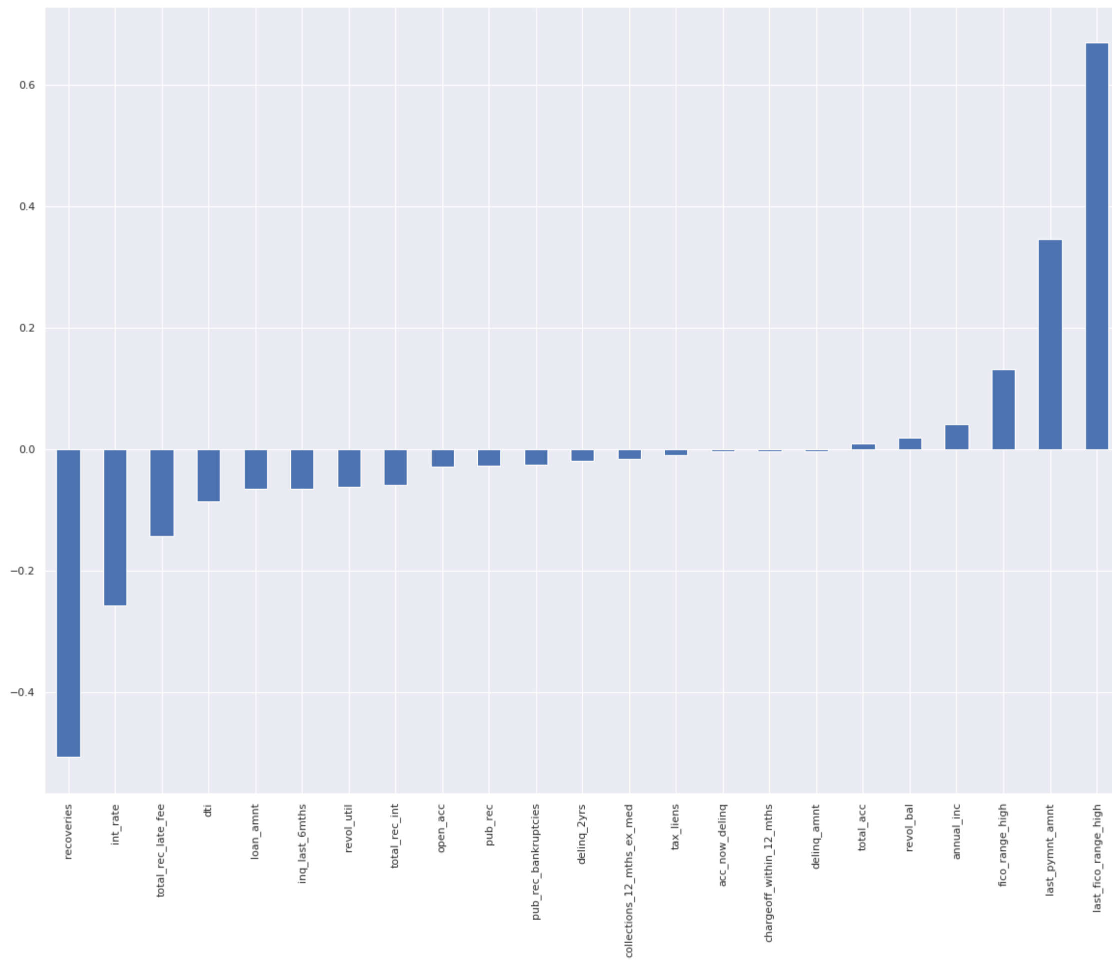
```
+------------+-------+
|loan_is_paid|  count|
```

```
+------------+-------+
|           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']
```

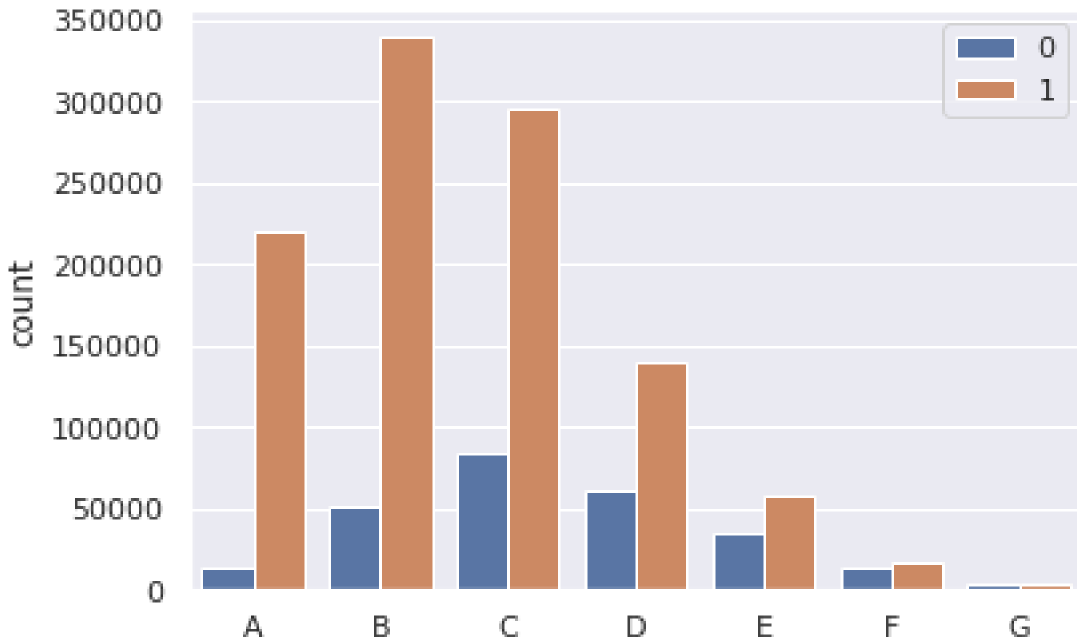
```
+-----------+
|term_months|
+-----------+
|         60|
|         36|
+-----------+
```

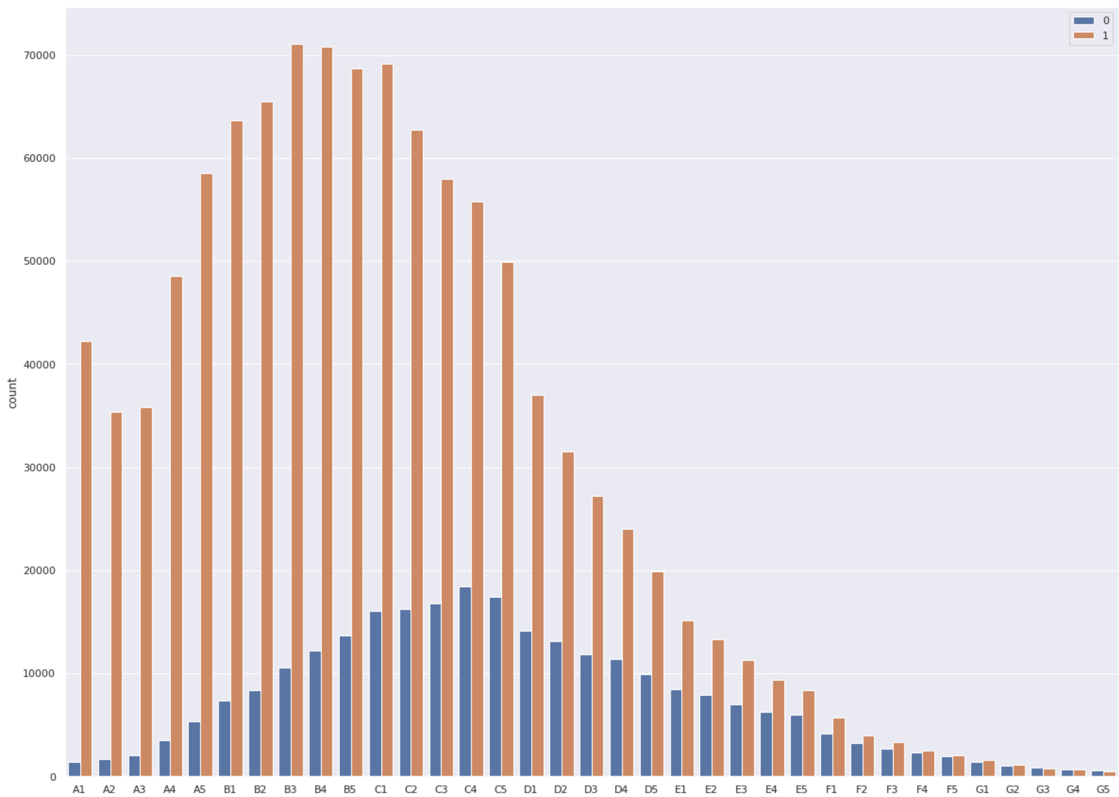
### 1.3.3 home_ownership Column

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






From the above graphs and analysis we can deduce that:

- 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

Schema:

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

First Row Of the dataframe:

```
+--------------------+------------+
|     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

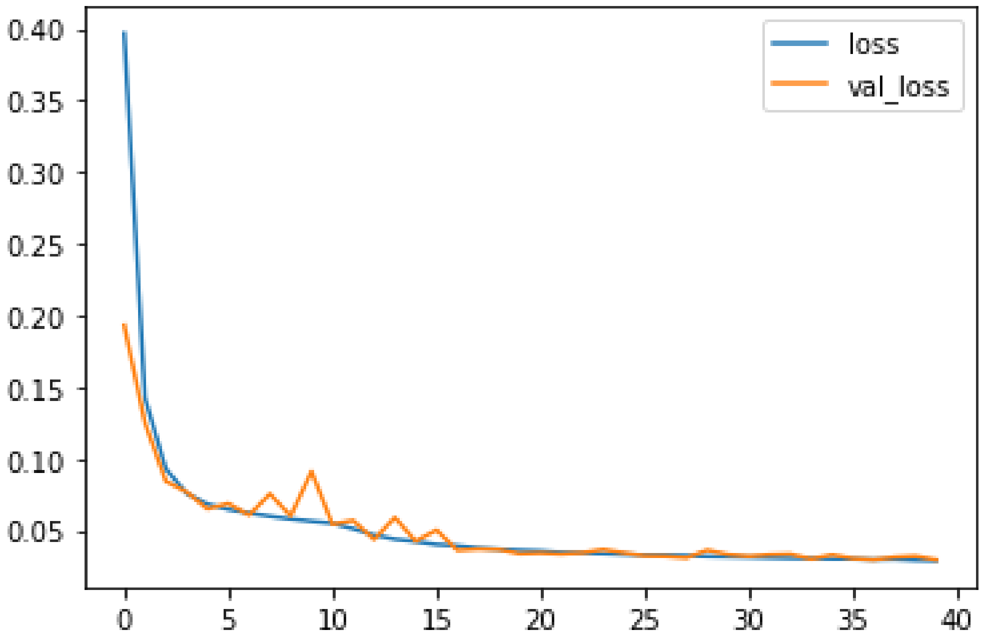
1. Build the following deeplearning model with this output shapes consists of 6 fully connected layer with relu function as the activation function and the final activation function is a sigmoid
2. Compile the model with adam optimizer with accuracy metrics and binary_crossentropy loss function

```
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
_________________________________________________________________
```

3. Transform to the dataframe to pandas dataframe before training
4. Split data 80% training set and 20% testing set
5. Finally, Train the model

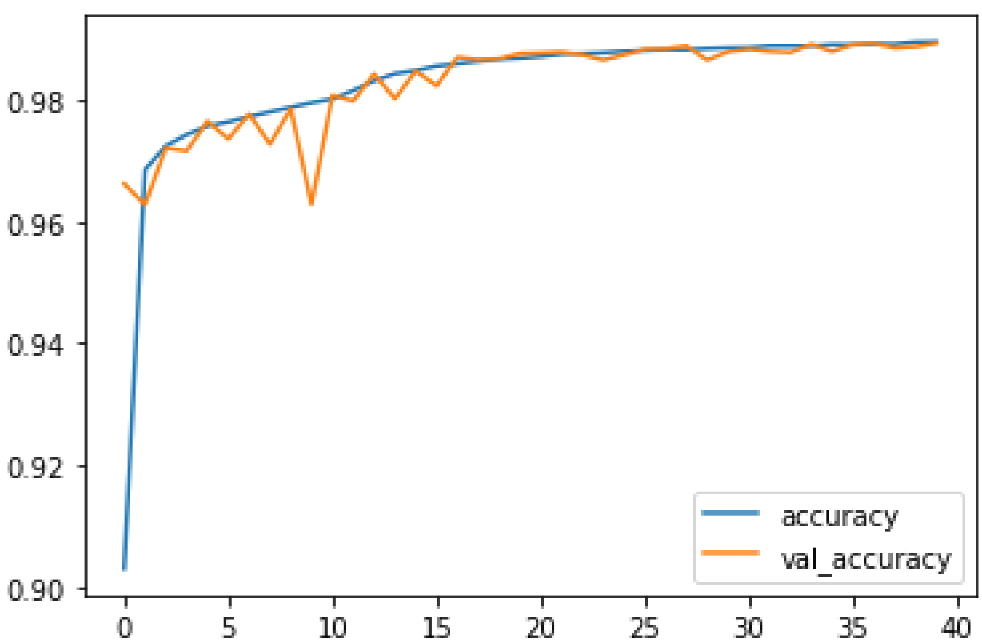
Loss Graph:

```
[71]: <AxesSubplot:>
```




Accuracy Graph:

```
[72]: <AxesSubplot:>
```



**Accuracy = 98.94%**

6. Saving the model ...

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

# Gradient Boosting Tree (spark)

1. Using the map reduce machine learning models of pyspark and spark dataframe we can build model using HDFS
2. Split data 80% training set and 20% testing set
3. Train the Gradient Boosting Tree Classifier using the training data
4. Finally, 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 False Positive, False Negative, True Positive and True Negative values

**Accuracy = 97.46%**

## 2 Conclusion

- We visualized the dataset using univarient and multivarient visualization and deduced some information from it
- We've done some Data cleaning and preprocessing to prepare the data to build our models
- We using HDFS for both data visualizing, data preprocessing and building gradient boosing tree classifier with accuracy 97.46%
- We also made a deeplearning model with accuracy 98.94%
- Both models assess whether or not a new customer is likely to pay back the loan, so this analysis is very important for decision making.