A Project Report on

Predictive Application Using Machine Learning Models Using Social Profile in Online P2P Lending Market

Work carried out for

Technocolabs Softwares Pvt. Ltd., Indore, Madhya Pradesh

Submitted to Pondicherry University in partial fulfilment of the requirement for the Award of the Degree of

Master of Business Administration

By

S. Suriya Pragash
(Reg. No. 21401039)

Under the Guidance of

Dr. R. Venkatesakumar, Professor
Department of Management Studies, Pondicherry University

&

YASIN SHAH, *CEO*,
Technocolabs Softwares Pvt. Ltd.

Department of Management Studies
Pondicherry University, Pondicherry, INDIA – 605 014

July – August 2022

DEPARTMENT OF MANAGEMENT STUDIES
SCHOOL OF MANAGEMENT
PONDICHERRY UNIVERSITY
PONDICHERRY-605014

CERTIFICATE

This is to certify that this project report entitled **"Predictive Application Using Machine Learning Models Using Social Profile in Online P2P Lending Market "** done for **Technocolabs Softwares Pvt. Ltd.,** is submitted by **SURIYA PRAGASH (Reg.No:21401039),** II MBA to the **DEPARTMENT OF MANAGEMENT STUDIES, SCHOOL OF MANAGEMENT, PONDICHERRY UNIVERSITY** in partial fulfilment of the requirements for the award of the degree of **MASTER OF BUSINESS ADMINISTRATION** and is a record of an original and bonafide work done under the guidance of **Dr. R. Venkatesakumar**, Professor, Department of Management Studies, Pondicherry University. This report has not formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title to the candidate and that the report represents an independent and original work on the part of the candidate.

Dr. R. Venkatesakumar
Professor and Project Guide
Department of Management Studies

Dr. B. CHARUMATHI
Professor and Head
Department of Management Studies

Date:
Place: Pondicherry 605 014

DECLARATION

I hereby declare that the project titled, **Predictive Application Using Machine Learning Models Using Social Profile in Online P2P Lending Market is** original work done by me under the guidance of **Dr. R. Venkatesakumar**, **Professor**, Department of Management Studies, Pondicherry University, and **YASIN SHAH, Technocolabs Softwares Pvt. Ltd., Indore, Madhya Pradesh**. This project or any part thereof has not been submitted for any Degree / Diploma / Associateship / Fellowship / any other similar title or recognition to this University or any other University.

I take full responsibility for the originality of this report. I am aware that I may have to forfeit the degree if plagiarism has been detected after the award of the degree. Notwithstanding the supervision provided to me by the Faculty Guide, I warrant that any alleged act(s) of plagiarism in this project report are entirely my responsibility. Pondicherry University and/or its employees shall under no circumstances whatsoever be under any liability of any kind in respect of the aforesaid act(s) of plagiarism.

SURIYA PRAGASH
21401039
II MBA
Department of Management Studies
Pondicherry University

Date:
Place: Puducherry

Acknowledgements

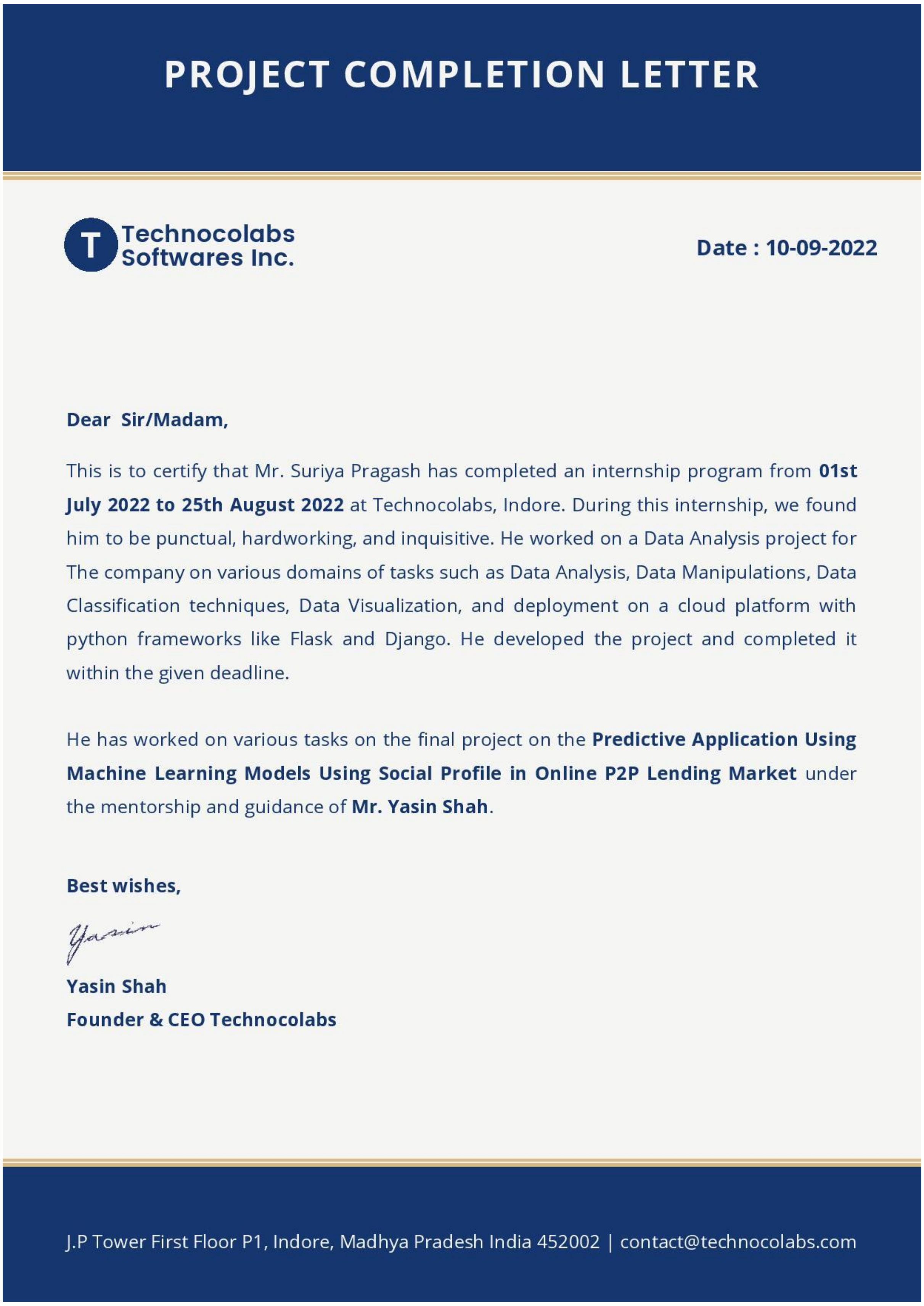
I thank my research guide Dr. R. Venkatesakumar, Department of Management Studies, Pondicherry University, Pondicherry for his incessant encouragement and support extended throughout the research period.

Thanks to Mr. Yasin Shah, Technocolabs Softwares, for sharing and discussion of various research related matters.

Many thanks to those faculty members who helped me in sharpening my thinking by cheerfully providing challenging comments and questions.

I thank my classmates of Pondicherry University who had given me moral support.

Foremost, I am grateful to my family who had to tolerate late night work, and curtailed weekends & vacations. The support and encouragement of my parents gave me the energy, stamina and inspiration to complete the project.



PROJECT COMPLETION LETTER

Technocolabs Softwares Inc.

Date : 10-09-2022

Dear Sir/Madam,

This is to certify that Mr. Suriya Pragash has completed an internship program from **01st July 2022 to 25th August 2022** at Technocolabs, Indore. During this internship, we found him to be punctual, hardworking, and inquisitive. He worked on a Data Analysis project for The company on various domains of tasks such as Data Analysis, Data Manipulations, Data Classification techniques, Data Visualization, and deployment on a cloud platform with python frameworks like Flask and Django. He developed the project and completed it within the given deadline.

He has worked on various tasks on the final project on the **Predictive Application Using Machine Learning Models Using Social Profile in Online P2P Lending Market** under the mentorship and guidance of **Mr. Yasin Shah**.

Best wishes,

Yasin

Yasin Shah
Founder & CEO Technocolabs

J.P Tower First Floor P1, Indore, Madhya Pradesh India 452002 | contact@technocolabs.com

Abstract

Lending Club Data Default Prediction Peer-to-peer lending is a relatively new form of credit that focuses on financing borrowers from their peers (small lenders) and individuals who earn interest on the money they lend. Borrowers can apply through an online platform for personal loans, often unsecured, that are financed by one or more peer investors. The P2P lender is not an actual lender, but rather an intermediary that facilitates the lending process and provides the platform. These platforms have developed in US, UK, Australia and other financial markets, but the US remains the leader in the peer-to-peer lending space. Peer-to-peer lending may not have begun in the US, but it has quickly spread to dominate the personal loan market and is slowly making its way into other markets. Lending Club, a San Francisco-based fintech company, works to facilitate peer-to-peer loans through their online lending platform. Started in 2007, their website allows individuals to publicly post loan applications, which other users can then browse and choose to fund. Their website makes its historical records publicly available, leading to the interesting question: can we determine an accurate way of using loan characteristics to predict which loans will be paid back in full and which will default? This project aims to develop a robust module for predicting your chances of loan approval. The features used for prediction are considered to be available at the time of loan origination and thus do not leak any information from the future.

About the Company

Technocolabs is an Indore, Madhya Pradesh based Start-up company. The company was established in 2019.The primary area of focus of the company is Machine Learning, Data Science and Artificial Intelligence based product development. The Chief Executive Officer of Technocolabs is Mr. Yasin Shah.

The technologies that they utilize are Django, Heroku, Java, Node JS, JS, Python, C, C++, C#, Android, React, SwiftUI, VUE.js, Angular, CSS and GIT.

The services that they are offer are in the domain of Machine Learning, Computer Vision, Mobile Application Development, Voice enabled Skills, Web Application Development, Big Data and Data Science.

Mission – "Our Mission is to enhance business growth of our customers with creative design, development and to deliver market defining high quality solutions that create value and reliable competitive advantage to customers around the globe."

Plan – "Our plan is to setup requirements according to our clients and customers satisfaction and proper understanding."

Vision – "We believe that we are on the face of the earth to make great products and that's not changing. We are constantly focusing on innovating. We believe in the simple not the complex. We believe that we need to own and control the primary technologies behind the products that we make and participate only in markets where we can make a significant contribution."

TABLE OF CONTENTS

Chapter – I

Introduction

LendingClub

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 (SEC), and to offer loan trading on a secondary market. LendingClub is the world's largest peer-to-peer lending platform.

Business Understanding:

LendingClub company which specialises in lending various types of loans to urban customers. When the company receives a loan application, the company has to make a decision for loan approval based on the applicant’s profile. Two types of risks are associated with the bank’s decision:

- If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company
- If the applicant is not likely to repay the loan, i.e., he/she is likely to default, then approving the loan may lead to a financial loss for the company

Using this data, it is possible to develop a default prediction model. Could we also use the data provided on the rejected loans to develop a stronger model? Are we underestimating the risk of certain segments by removing the data from the riskiest applications? The challenge is that, since these riskiest loans were never booked, whether the loan would have defaulted or not is unknown.

In industry, there is no consensus on how to manage this situation. Some well-established firms choose to ignore this issue, while many large lenders and credit bureau anciencies, as well as FICO, use proprietary techniques to adjust this potential bias via Reject Inference.

The data given contains the information about past loan applicants and whether they ‘defaulted’ or not. The aim is to identify patterns which indicate if a person is likely to default, which may be used for takin actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc.

When a person applies for a loan, there are two types of decisions that could be taken by the company:

1. Loan accepted: If the company approves the loan, there are 3 possible scenarios described below:

- Fully paid: Applicant has fully paid the loan (the principal and the interest rate)
- Current: Applicant is in the process of paying the instalments, i.e., the tenure of the loan is not yet completed. These candidates are not labelled as 'defaulted'.
- Charged-off: Applicant has not paid the instalments in due time for a long period of time, i.e., he/she has defaulted on the loan

2. Loan rejected: The company had rejected the loan (because the candidate does not meet their requirements etc.). Since the loan was rejected, there is no transactional history of those applicants with the company and so this data is not available with the company (and thus in this dataset)

Chapter –II

Objective

- LendingClub is the largest online loan marketplace, facilitating personal loans, business loans, and financing of medical procedures. Borrowers can easily access lower interest rate loans through a fast online interface.
- Like most other lending companies, lending loans to ‘risky’ applicants is the largest source of financial loss (called credit loss). The credit loss is the amount of money lost by the lender when the borrower refuses to pay or runs away with the money owed. In other words, borrowers who defaultcause the largest amount of loss to the lenders. In this case, the customers labelled as 'charged-off' or the 'defaulters'.
- If one is able to identify these risky loan applicants, then such loans can be reduced thereby cutting down the amount of credit loss. Identification of such applicants using EDA and machine learning is the aim of this case study.
- In other words, the company wants to understand the driving factors (or driver variables) behind loan default, i.e. the variables which are strong indicators of default. The company can utilise this knowledge for its portfolio and risk assessment.

The core question for LendingClub becomes "Who can repay their loans?". To support this question, we first ask several other questions:

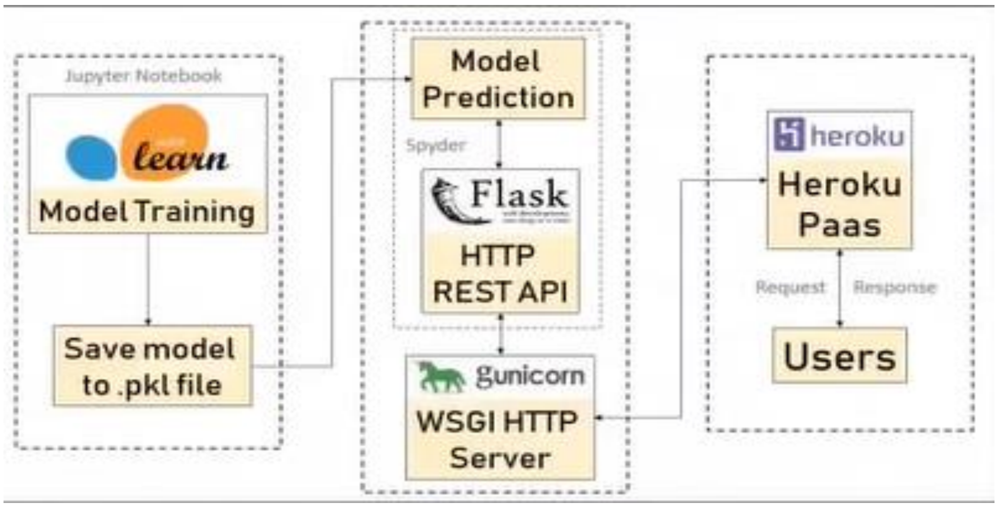
- Who is getting loans?
- What kind of loans are being given out?
- How accurate is LendingClubs loan grading system?
- What data points are important in determining if someone will repay?

After answering these questions, we explore making accurate machine learning algorithms and attempt to answer our core question.

Chapter –III

Methodology

Design Architecture of the Website



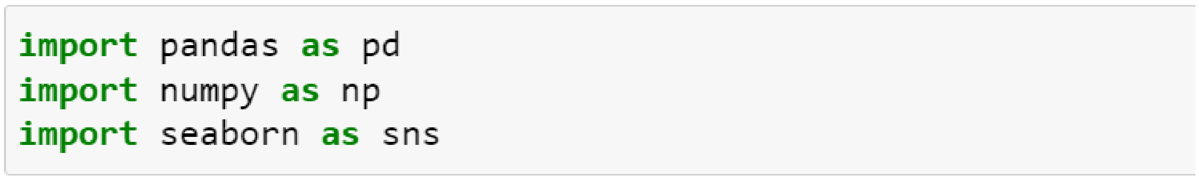
The first vertical includes the data exploration, data cleaning, feature engineering, model training, model validation, model selection hyperparameter tuning and finally saving it into a pickle file. All these data analysis steps are carried out in the Jupyter notebook using the scikit learn library. This pickle file contains the final model which has been used in app(.py) file to load the model and generate prediction from the input data that will be received from the html form. This app file is a flask web framework which provides tools, libraries and technologies that allows to build a web application. This was written in python using Visual Studio and all template (html, .CSS) files for the webpages were included in the folders along with the app file.
Using get http method, data was transferred to the loaded model and the result was posted back to the web page using the post http method. This application was then deployed on the Heroku server which uses gunicorn WSGI HTTP Server. Heroku is a free platform which provides 5 free apps to be deployed on their servers. It gets linked with your GitHub account so, this application was first uploaded on GitHub and then the repository was deployed on Heroku servers. This is the entire structure of the web app created to predict the loan defaults. In further sections, the data science and analysis part will be discussed.

Chapter –IV

Data Preprocessing

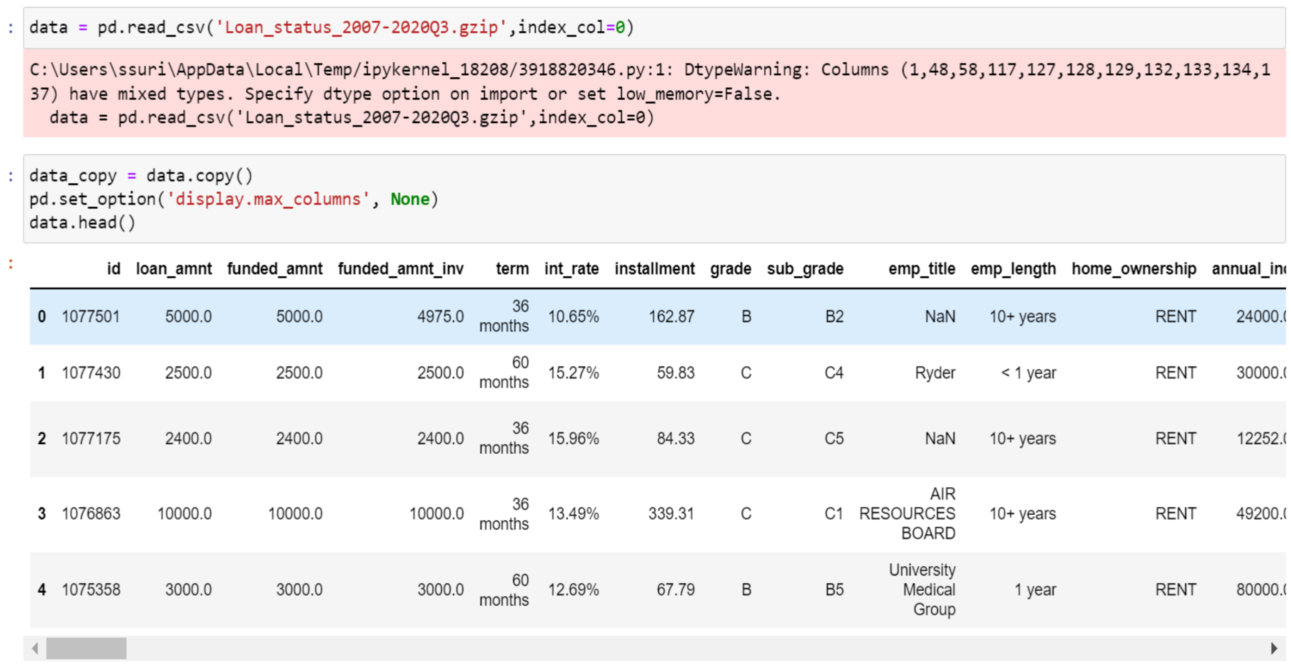
The dataset used for this project was available on the LendingClub website. This dataset contains completed load data for all loans issued through the 2007-2021, including the current loan status and latest payment information. This data set is a matrix of about 3000000 observations and 141 features. From that dataset we are going to extract data last 4 years data. Basically, it's a P2P dataset so it's having all the information about borrowers as well as investors. But we are going to build model on defaulters loan prediction. So, we extract features related to borrowers and save that as a csv file.

Importing Libraries:



```
import pandas as pd
import numpy as np
import seaborn as sns
```

Import dataset:



```
: data = pd.read_csv('Loan_status_2007-2020Q3.gzip',index_col=0)
```

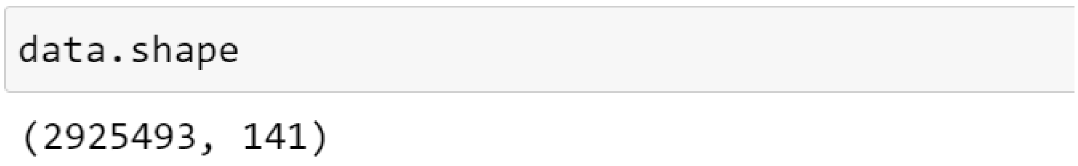
```
C:\Users\ssuri\AppData\Local\Temp/ipykernel_18208/3918820346.py:1: DtypeWarning: Columns (1,48,58,117,127,128,129,132,133,134,1
37) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv('Loan_status_2007-2020Q3.gzip',index_col=0)
```

```
: data_copy = data.copy()
pd.set_option('display.max_columns', None)
data.head()
```

:

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_in
0	1077501	5000.0	5000.0	4975.0	36 months	10.65%	162.87	B	B2	NaN	10+ years	RENT	24000.
1	1077430	2500.0	2500.0	2500.0	60 months	15.27%	59.83	C	C4	Ryder	< 1 year	RENT	30000.
2	1077175	2400.0	2400.0	2400.0	36 months	15.96%	84.33	C	C5	NaN	10+ years	RENT	12252.
3	1076863	10000.0	10000.0	10000.0	36 months	13.49%	339.31	C	C1	AIR RESOURCES BOARD	10+ years	RENT	49200.
4	1075358	3000.0	3000.0	3000.0	60 months	12.69%	67.79	B	B5	University Medical Group	1 year	RENT	80000.

Data Shape:



```
data.shape
```

```
(2925493, 141)
```

Target Variable:

```
# We want to predict if a particular loan was fully paid or charged off
#(if it was paid or the customer was not able to pay back the loan amount)
data['loan_status'].value_counts()
```

```
Fully Paid                                             1497783
Current                                                1031016
Charged Off                                             362548
Late (31-120 days)                                       16154
In Grace Period                                          10028
Late (16-30 days)                                         2719
Issued                                                    2062
Does not meet the credit policy. Status:Fully Paid        1988
Does not meet the credit policy. Status:Charged Off        761
Default                                                    433
Name: loan_status, dtype: int64
```

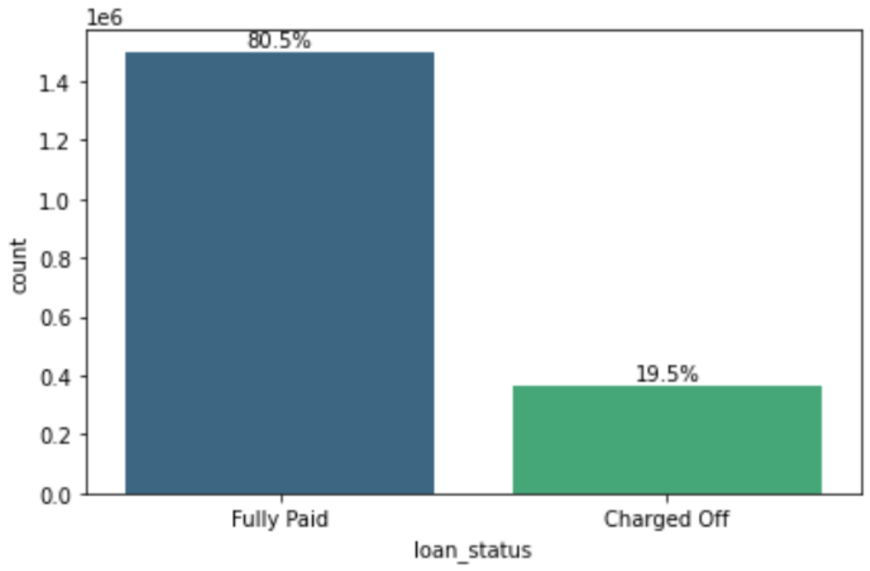
This dataset was having information about people who fully paid their loan, default, charged off and then people whose loan status is current. From this we are going to take Fully Paid and Charged off. Dropping all other values except Fully Paid and Charged off.

```
# We will drop rows which have loan_status other than Fully Paid and Charged Off
data = data[(data['loan_status'] == 'Fully Paid') | (data['loan_status'] == 'Charged Off')]
```

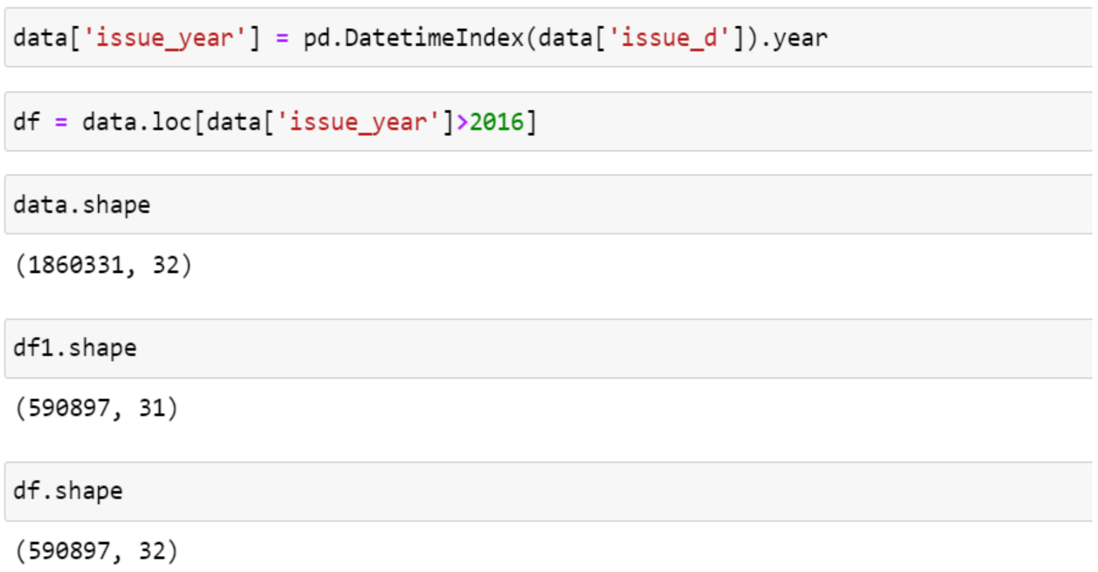
```
data.shape
```

```
(1860331, 141)
```

```
# lets check if loan_status now has only fully paid and charged off
ax=sns.countplot(x=data['loan_status'], data=data, palette='viridis')
percent_value()
```



Extract the feature which were having borrowers information and loans that was given after 2016.



```
data['issue_year'] = pd.DatetimeIndex(data['issue_d']).year
```

```
df = data.loc[data['issue_year']>2016]
```

```
data.shape
```

```
(1860331, 32)
```

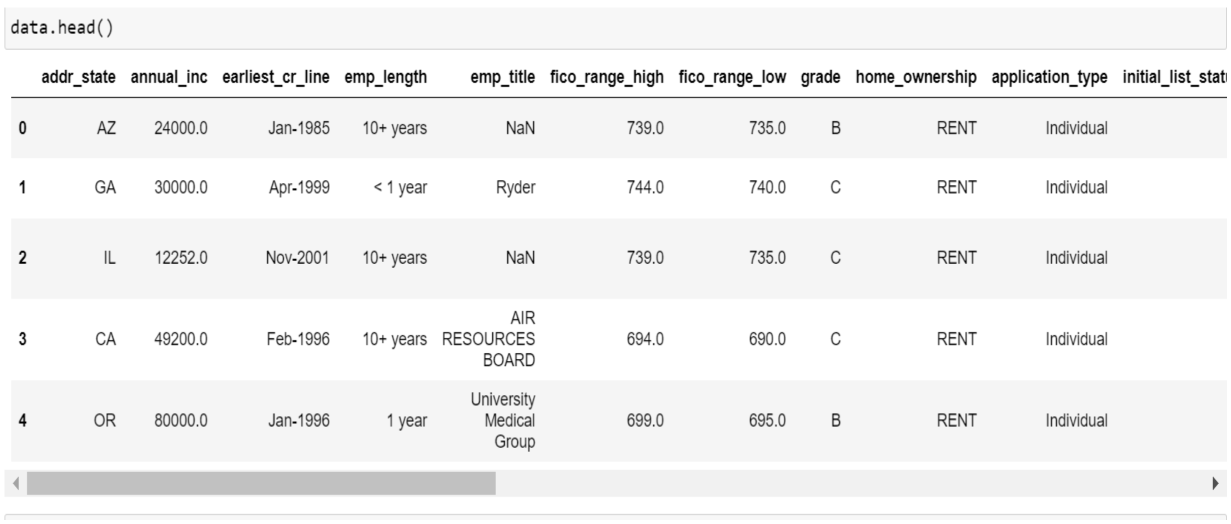
```
df1.shape
```

```
(590897, 31)
```

```
df.shape
```

```
(590897, 32)
```

For our further analysis we are using the borrowers data who got loan after 2016



```
data.head()
```

	addr_state	annual_inc	earliest_cr_line	emp_length	emp_title	fico_range_high	fico_range_low	grade	home_ownership	application_type	initial_list_stat
0	AZ	24000.0	Jan-1985	10+ years	NaN	739.0	735.0	B	RENT	Individual	
1	GA	30000.0	Apr-1999	< 1 year	Ryder	744.0	740.0	C	RENT	Individual	
2	IL	12252.0	Nov-2001	10+ years	NaN	739.0	735.0	C	RENT	Individual	
3	CA	49200.0	Feb-1996	10+ years	AIR RESOURCES BOARD	694.0	690.0	C	RENT	Individual	
4	OR	80000.0	Jan-1996	1 year	University Medical Group	699.0	695.0	B	RENT	Individual	

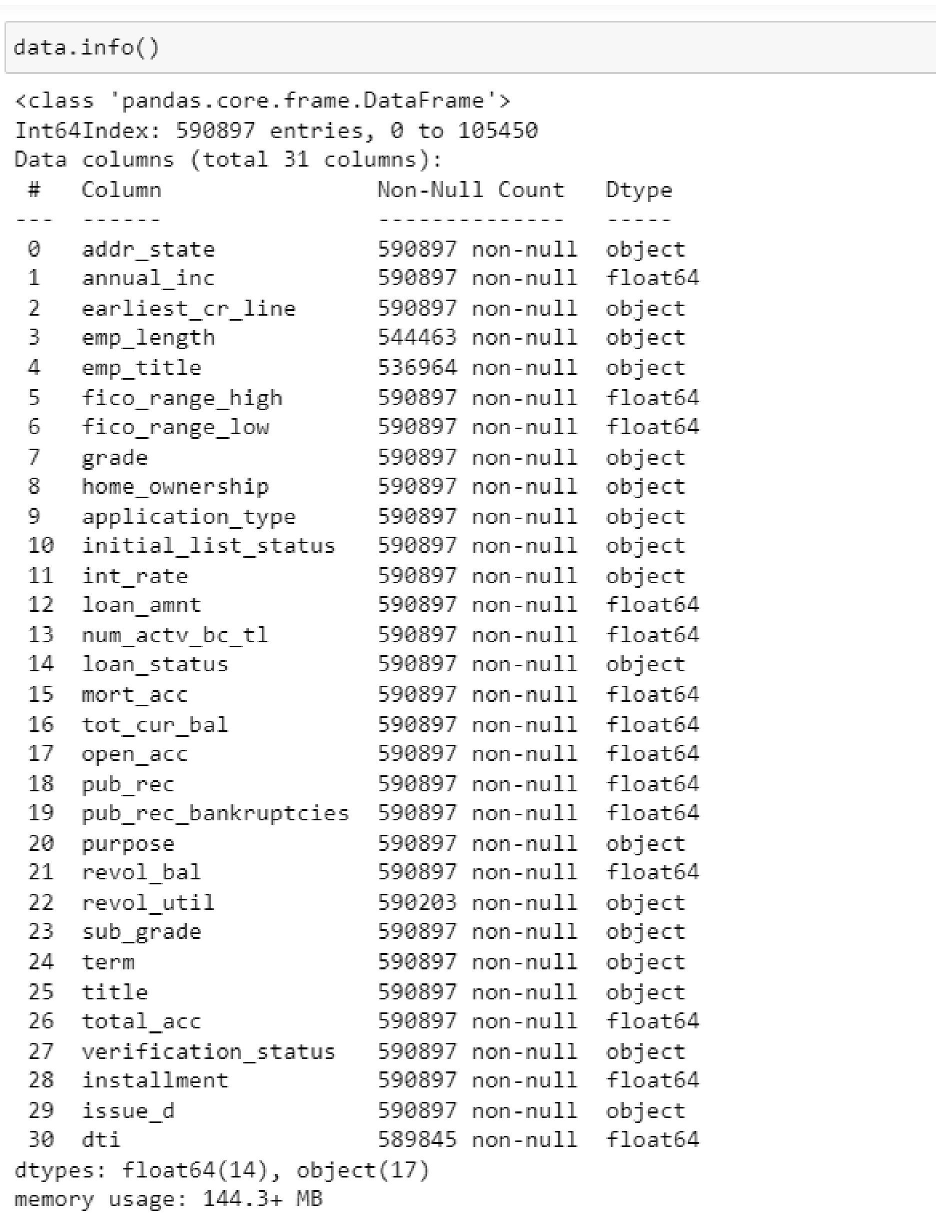
In the above table it shows the first 5 rows of the data

Handling Missing Values:

```
# percentage of data missing
((data.isnull().sum()/len(data))*100).sort_values(ascending=False)
```

```
emp_title               9.127310
emp_length              7.858222
dti                     0.178034
revol_util              0.117449
open_acc                0.000000
issue_d                 0.000000
installment             0.000000
verification_status     0.000000
total_acc               0.000000
title                   0.000000
term                    0.000000
sub_grade               0.000000
revol_bal               0.000000
purpose                 0.000000
pub_rec_bankruptcies    0.000000
pub_rec                 0.000000
addr_state              0.000000
tot_cur_bal             0.000000
annual_inc              0.000000
loan_status             0.000000
num_actv_bc_tl          0.000000
loan_amnt               0.000000
int_rate                0.000000
initial_list_status     0.000000
application_type        0.000000
home_ownership          0.000000
grade                   0.000000
fico_range_low          0.000000
fico_range_high         0.000000
earliest_cr_line        0.000000
mort_acc                0.000000
dtype: float64
```

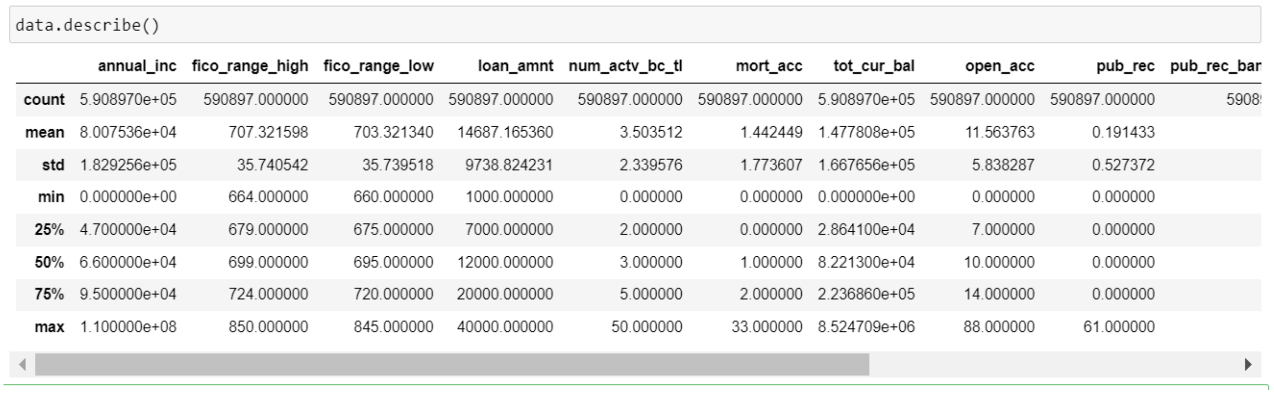
Data Info:



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 590897 entries, 0 to 105450
Data columns (total 31 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   addr_state            590897 non-null  object 
 1   annual_inc            590897 non-null  float64
 2   earliest_cr_line      590897 non-null  object 
 3   emp_length            544463 non-null  object 
 4   emp_title             536964 non-null  object 
 5   fico_range_high       590897 non-null  float64
 6   fico_range_low        590897 non-null  float64
 7   grade                 590897 non-null  object 
 8   home_ownership        590897 non-null  object 
 9   application_type      590897 non-null  object 
 10  initial_list_status   590897 non-null  object 
 11  int_rate              590897 non-null  object 
 12  loan_amnt             590897 non-null  float64
 13  num_actv_bc_tl        590897 non-null  float64
 14  loan_status           590897 non-null  object 
 15  mort_acc              590897 non-null  float64
 16  tot_cur_bal           590897 non-null  float64
 17  open_acc              590897 non-null  float64
 18  pub_rec               590897 non-null  float64
 19  pub_rec_bankruptcies  590897 non-null  float64
 20  purpose               590897 non-null  object 
 21  revol_bal             590897 non-null  float64
 22  revol_util            590203 non-null  object 
 23  sub_grade             590897 non-null  object 
 24  term                  590897 non-null  object 
 25  title                 590897 non-null  object 
 26  total_acc             590897 non-null  float64
 27  verification_status   590897 non-null  object 
 28  installment           590897 non-null  float64
 29  issue_d               590897 non-null  object 
 30  dti                   589845 non-null  float64
dtypes: float64(14), object(17)
memory usage: 144.3+ MB
```

Data describe:



```
data.describe()
```

	annual_inc	fico_range_high	fico_range_low	loan_amnt	num_actv_bc_tl	mort_acc	tot_cur_bal	open_acc	pub_rec	pub_rec_bar
count	5.908970e+05	590897.000000	590897.000000	590897.000000	590897.000000	590897.000000	5.908970e+05	590897.000000	590897.000000	5908
mean	8.007536e+04	707.321598	703.321340	14687.165360	3.503512	1.442449	1.477808e+05	11.563763	0.191433	
std	1.829256e+05	35.740542	35.739518	9738.824231	2.339576	1.773607	1.667656e+05	5.838287	0.527372	
min	0.000000e+00	664.000000	660.000000	1000.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	
25%	4.700000e+04	679.000000	675.000000	7000.000000	2.000000	0.000000	2.864100e+04	7.000000	0.000000	
50%	6.600000e+04	699.000000	695.000000	12000.000000	3.000000	1.000000	8.221300e+04	10.000000	0.000000	
75%	9.500000e+04	724.000000	720.000000	20000.000000	5.000000	2.000000	2.236860e+05	14.000000	0.000000	
max	1.100000e+08	850.000000	845.000000	40000.000000	50.000000	33.000000	8.524709e+06	88.000000	61.000000	

Chapter –V

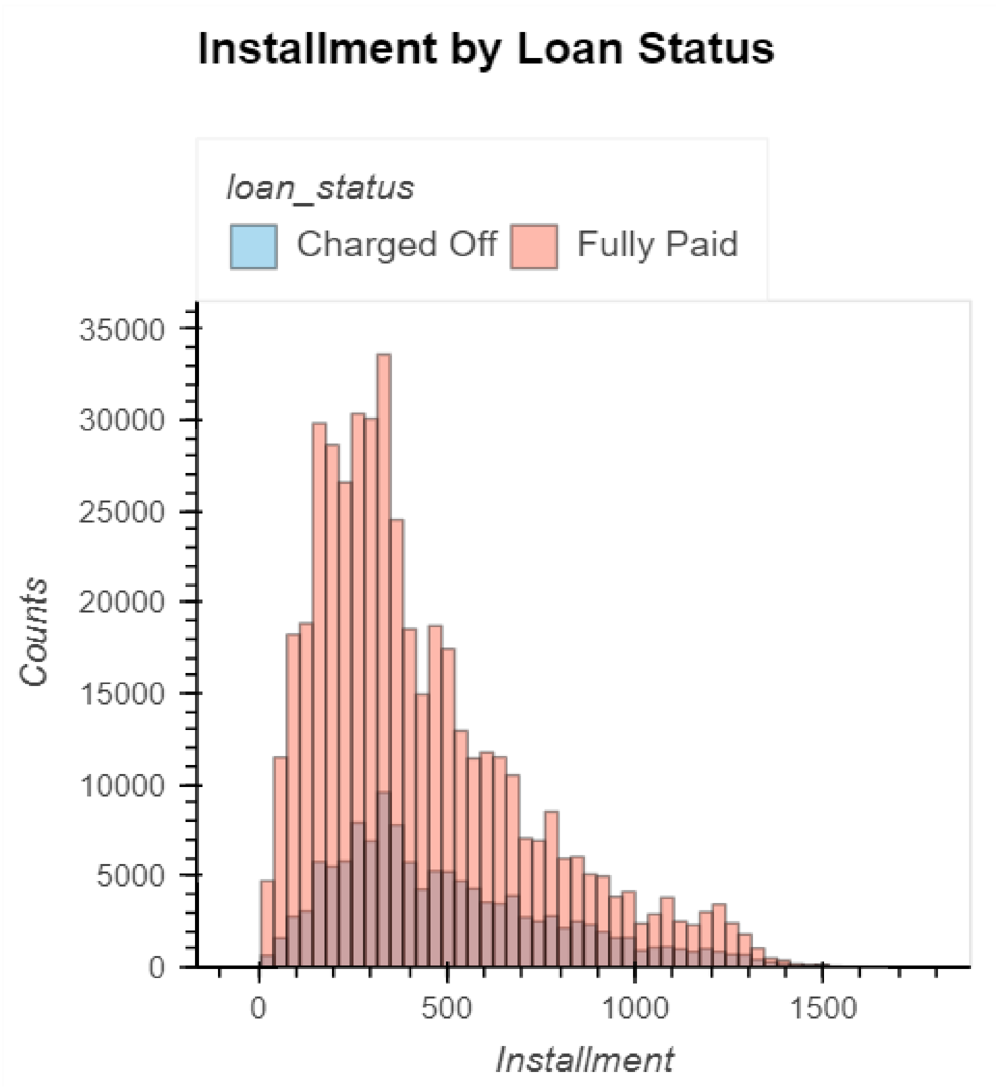
Exploratory Data Analysis

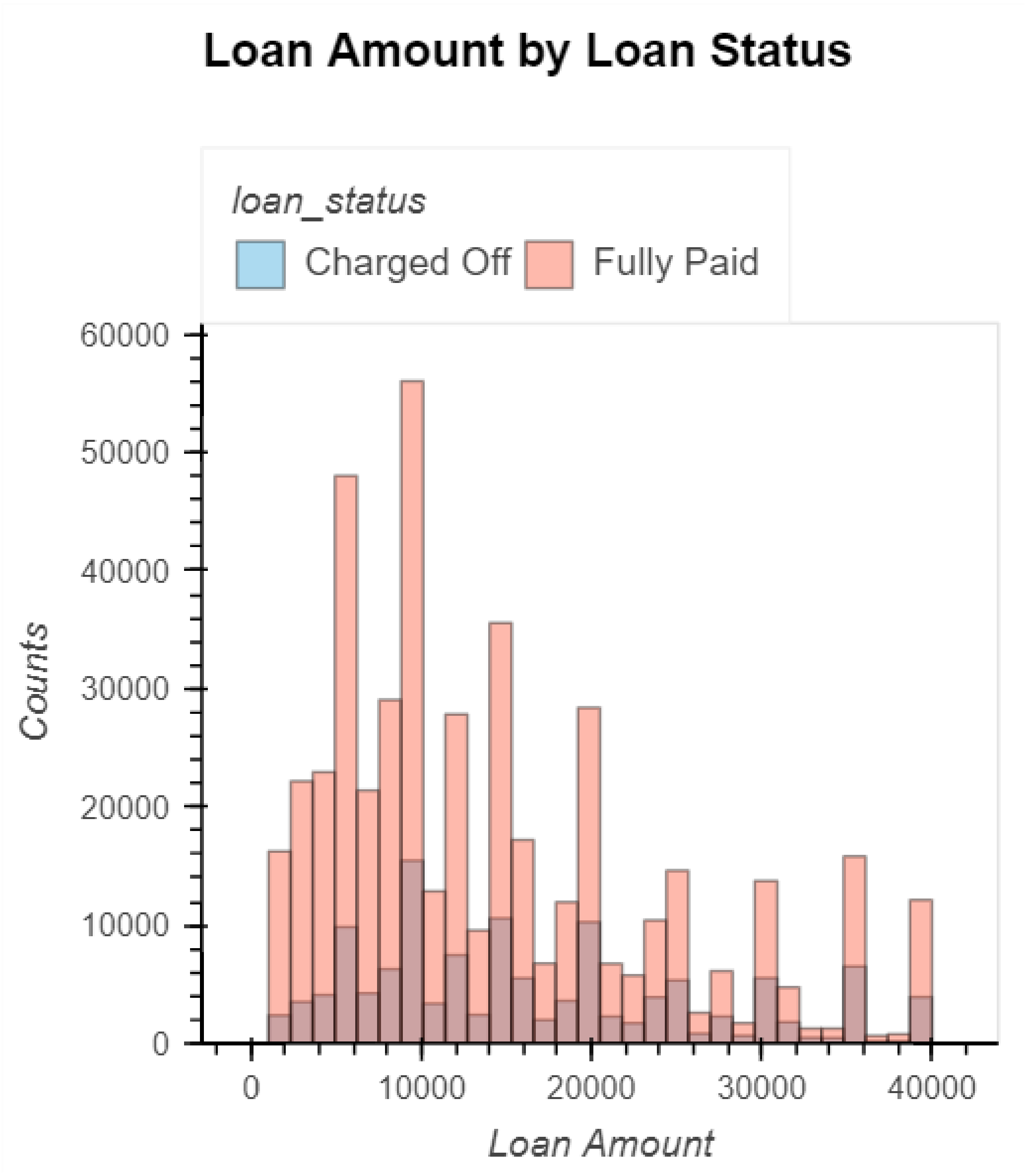
Exploratory Data Analysis, or EDA, is an important step in any Data Analysis or Data Science project. EDA is the process of investigating the dataset to discover patterns, and anomalies (outliers), and form hypotheses based on our understanding of the dataset.

EDA involves generating summary statistics for numerical data in the dataset and creating various graphical representations to understand the data better.

Loan_amnt & Installment:

- **Installment**: The monthly payment owed by the borrower if the loan originates.
- **Loan_amnt**: The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.

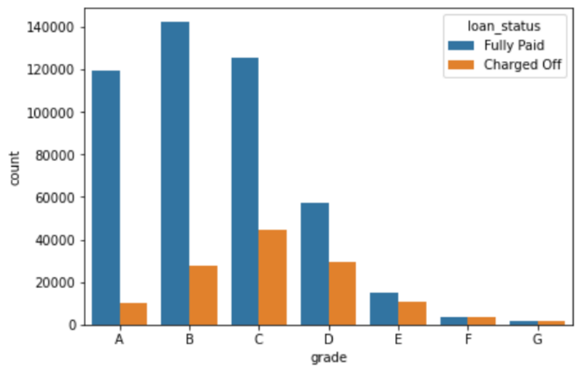


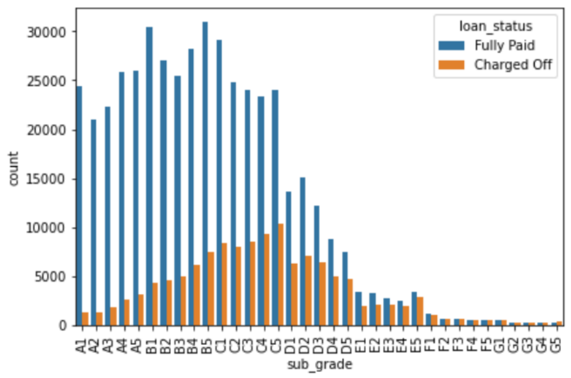


If the loan amount and installment is lesser then majority of people paid it fully.

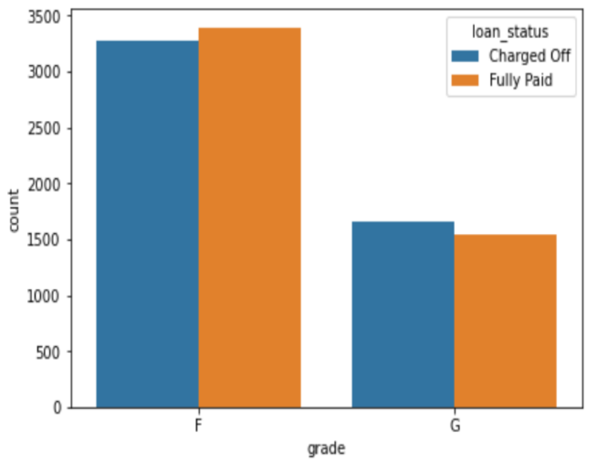
- For installment, the charged off rate is higher in the range of 0 to 500 but we can't conclude anything, so we check their annual income and their grade for better understanding.
- For loan amount, the higher the amount more the people likely to be charged off.

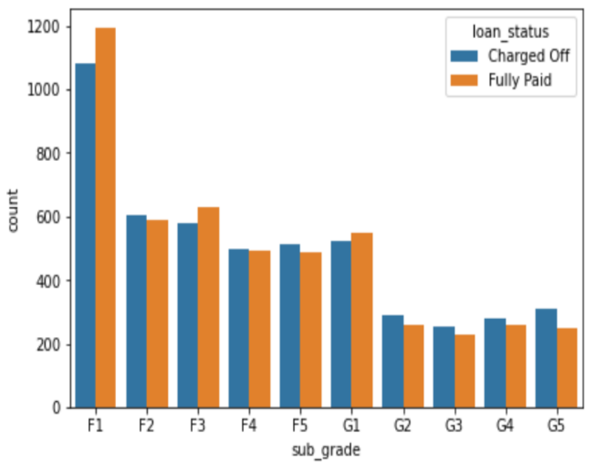
Grade & Sub Grade:





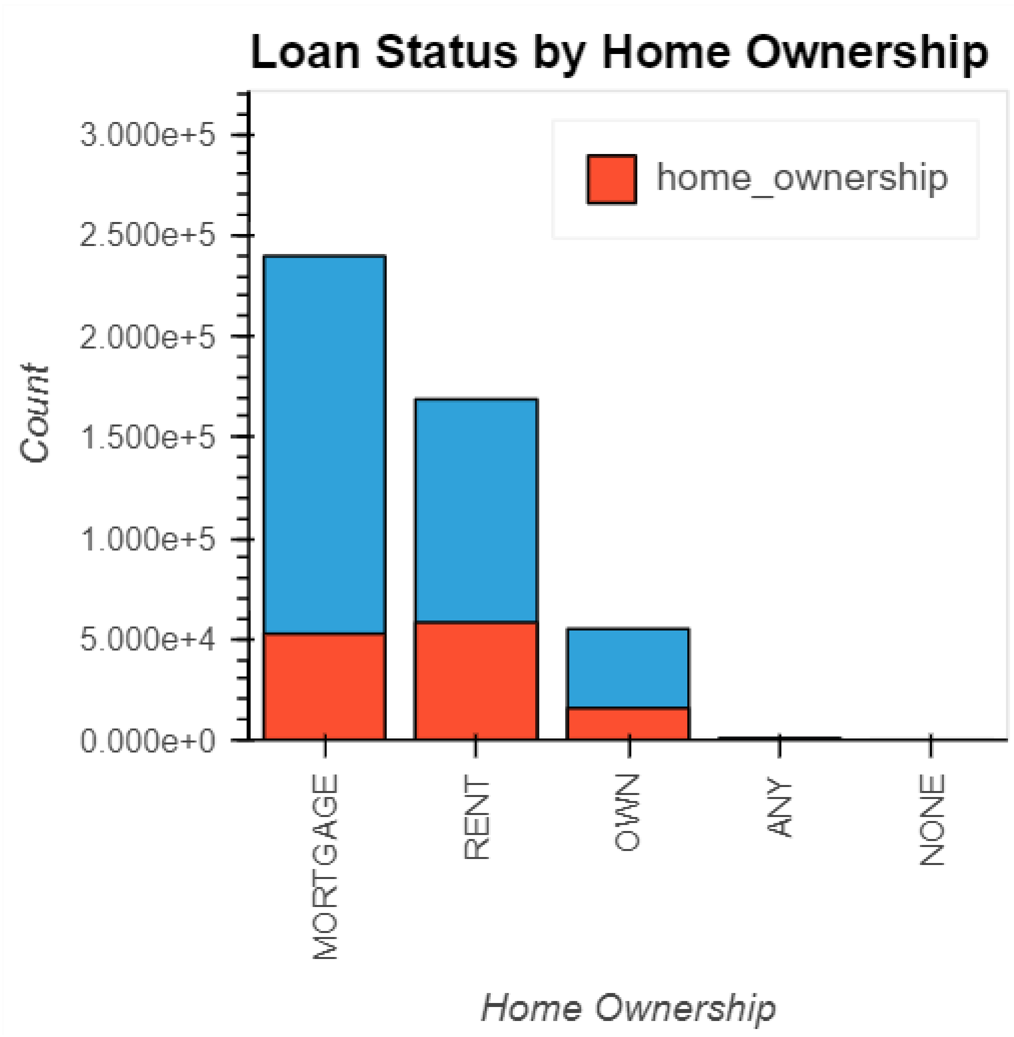
- Borrowers who are having grade B is mostly paid their loan and highest number of charged off is from grade C.
- It looks like F and G subgrades don't get paid back that often. Isolate those and recreate the countplot just for those subgrades.



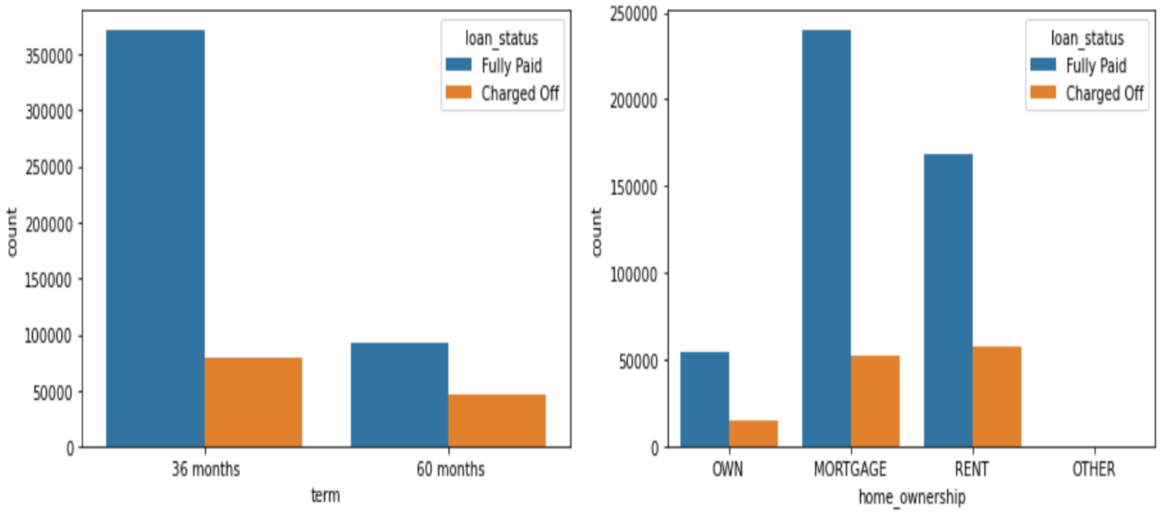


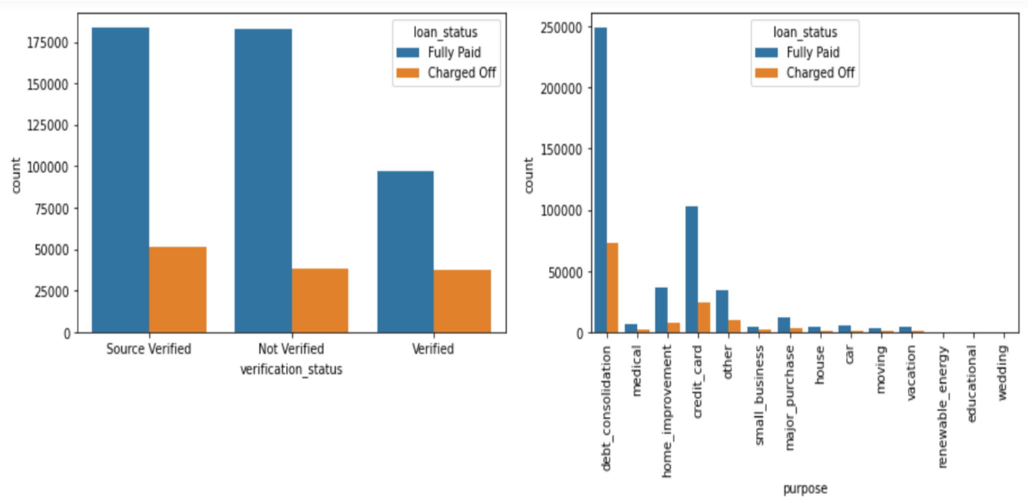
Term, Home ownership, Verification Status & Purpose:

- Term: The number of payments on the loan. Values are in months and can be either 36 or 60.
- Home Ownership: The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are RENT, OWN, MORTGAGE, OTHER
- Verification Status: Indicates if income was verified by LC, not verified, or if the income source was verified
- Purpose: A category provided by the borrower for the loan request.



People whose home ownership category are rent and mortgage are more likely to be charged off

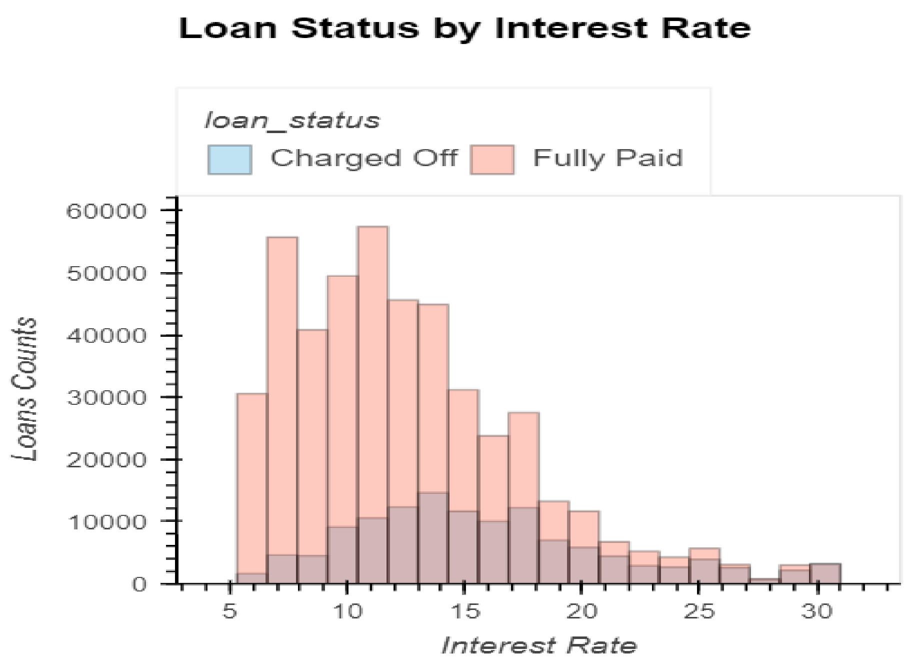




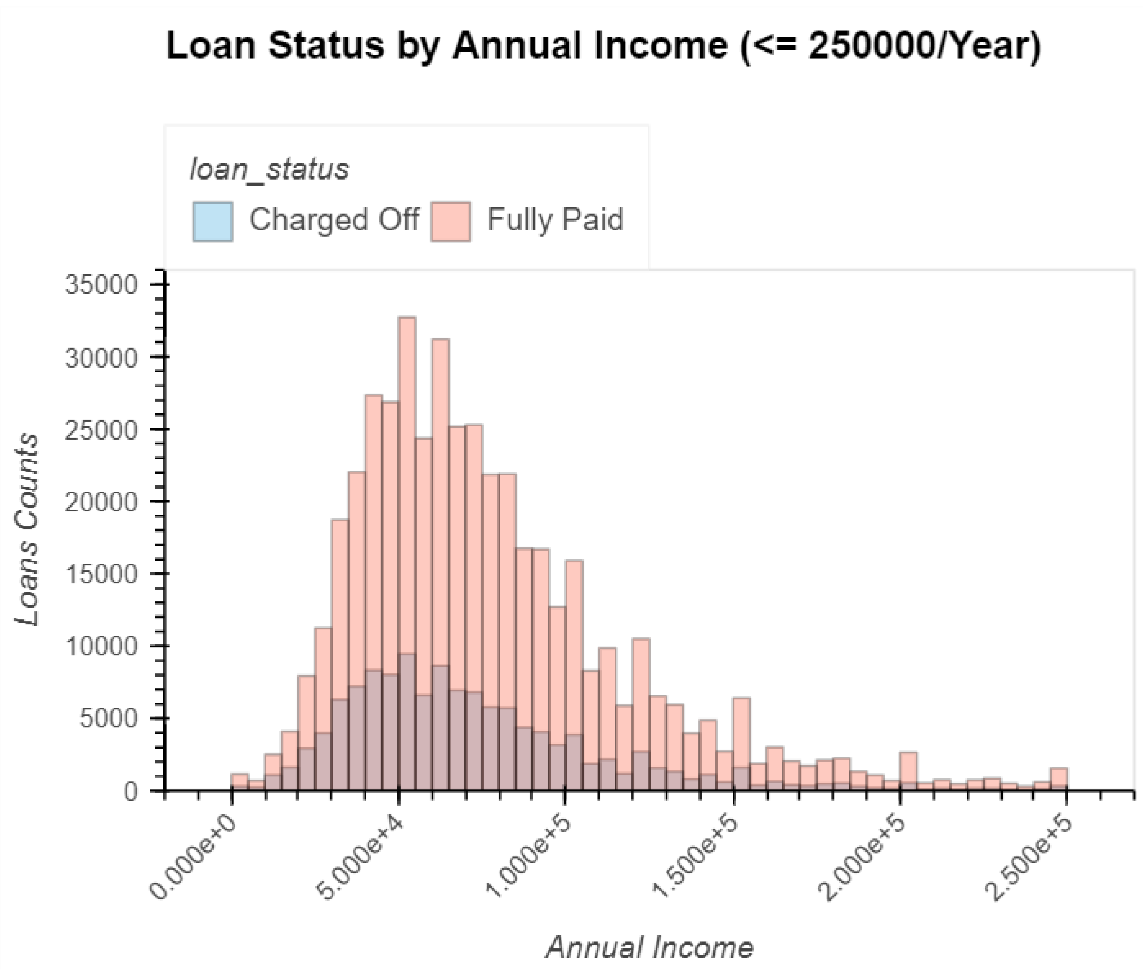
- Borrowers whose income source was verified are more likely to be charged off. So, there is an issue with companies verification they need to change that, if it continues it'll impact and more borrowers will be defaulters.
- Majority of borrowers whose purpose was debt consolidation were didn't paid their loan. So, company need to focus more on that kind of borrowers.

Interest Rate& Annual Income

- **Interest Rate**: Interest Rate on the loan
- **Annual Income**: The self-reported annual income provided by the borrower during registration



Number of Charged off is higher when the interest rate is higher, When the interest rate is less majority of borrowers fully paid their loan.



- When annual income is hight the number of borrowers who paid their loan is also high.
- Borrower whose salary is less they were not paid their loan properly. So, company need to approve their loan carefully or else give them lesser loan amount and high interest rate.

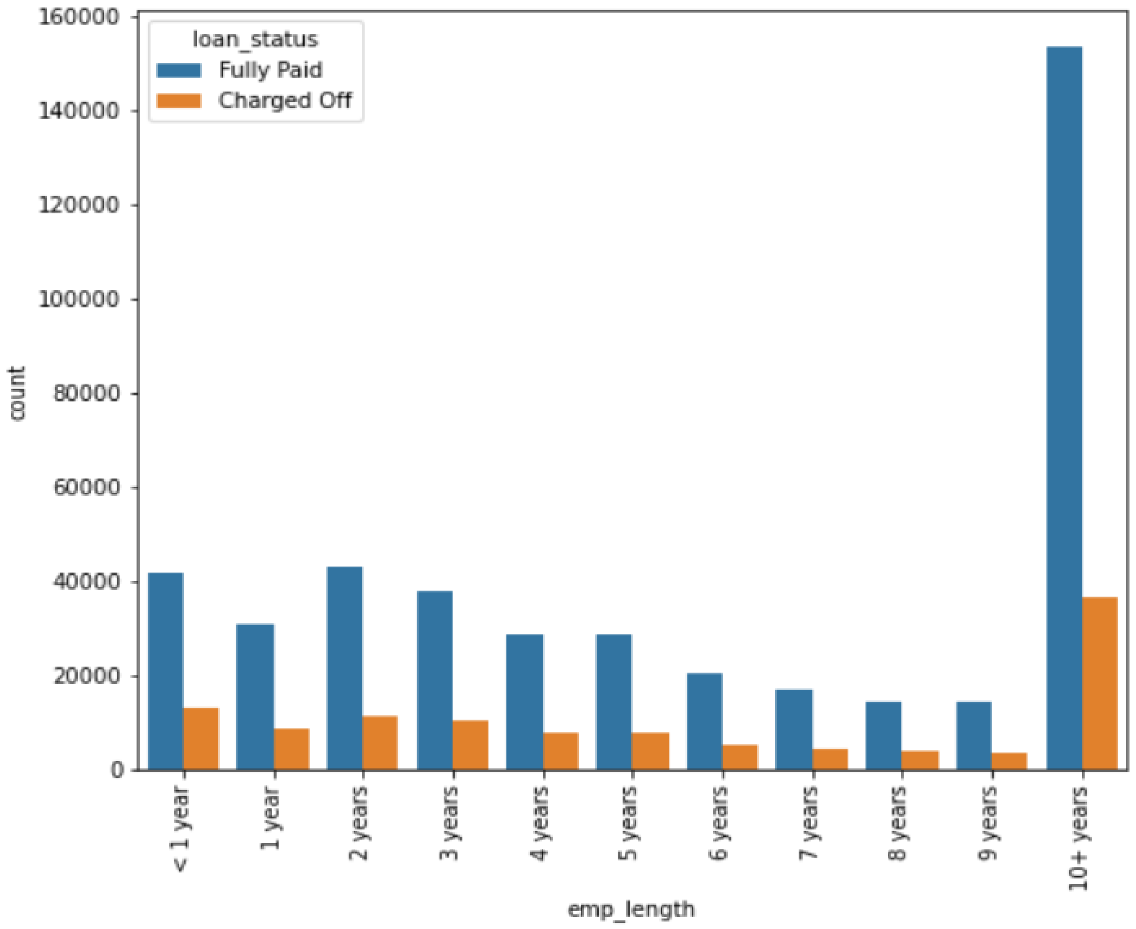
```
data.groupby(by='loan_status')['annual_inc'].describe()
```

loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	126758.0	75019.116017	316037.474357	0.0	44000.0	63000.0	90000.0	110000000.0
Fully Paid	464139.0	81456.241297	123750.075420	0.0	48000.0	68000.0	98000.0	61000000.0

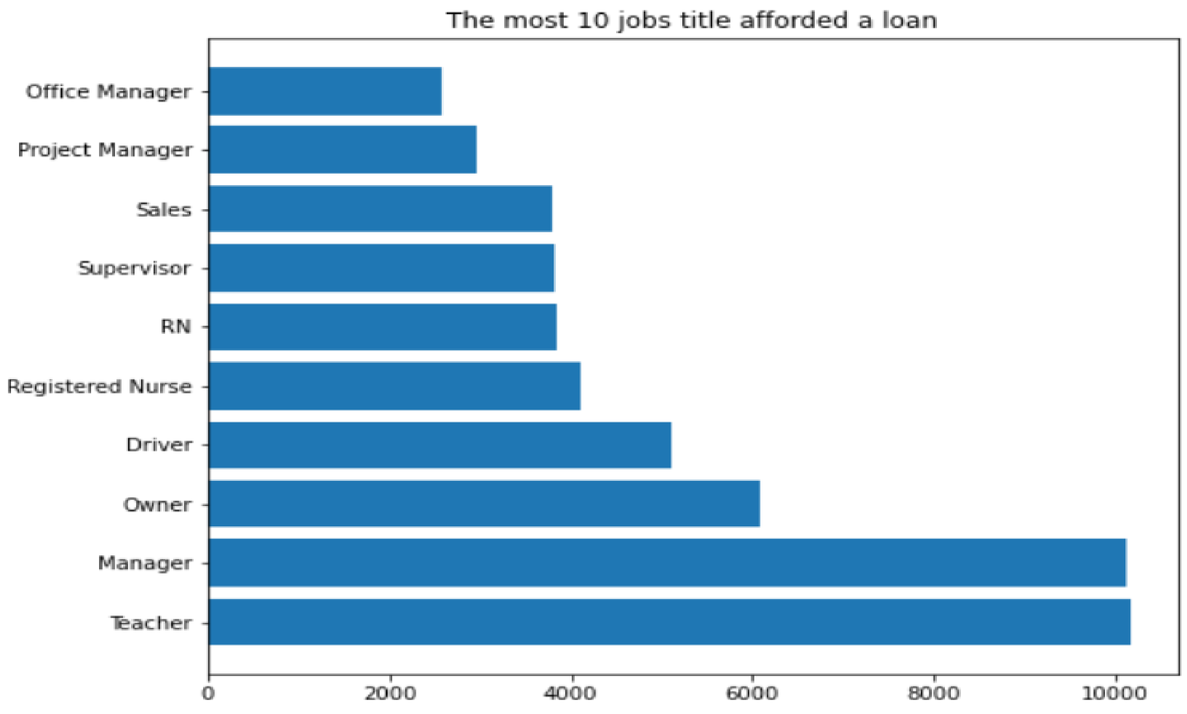
Maximum annual income of charged off category people is 110000000 which is an outlier, so we need to remove outliers before train our model.

Employee title & Employment length

Employee title: The job title supplied by the Borrower when applying for the loan.
Employment length: Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.



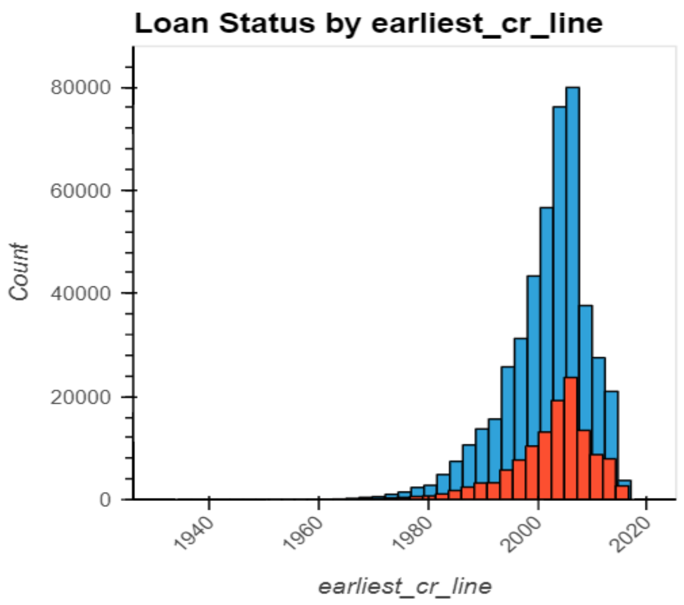
Majority of loan was given to the people who were having 10+ years employment length.

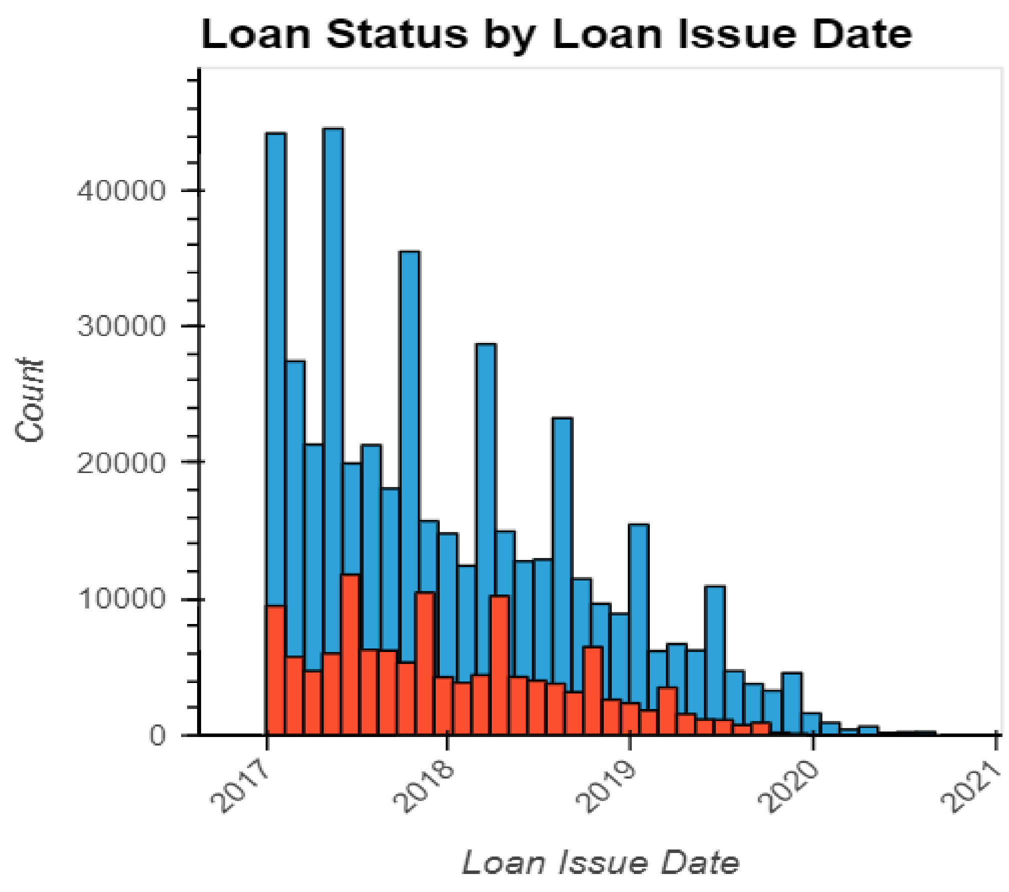


- Most of the borrowers were teachers then followed by manager and owner.

Issued Date, Earliest Credit Line:

- Issue date: The month which the loan was funded
- Earliest credit line: The month the borrower's earliest reported credit line was opened

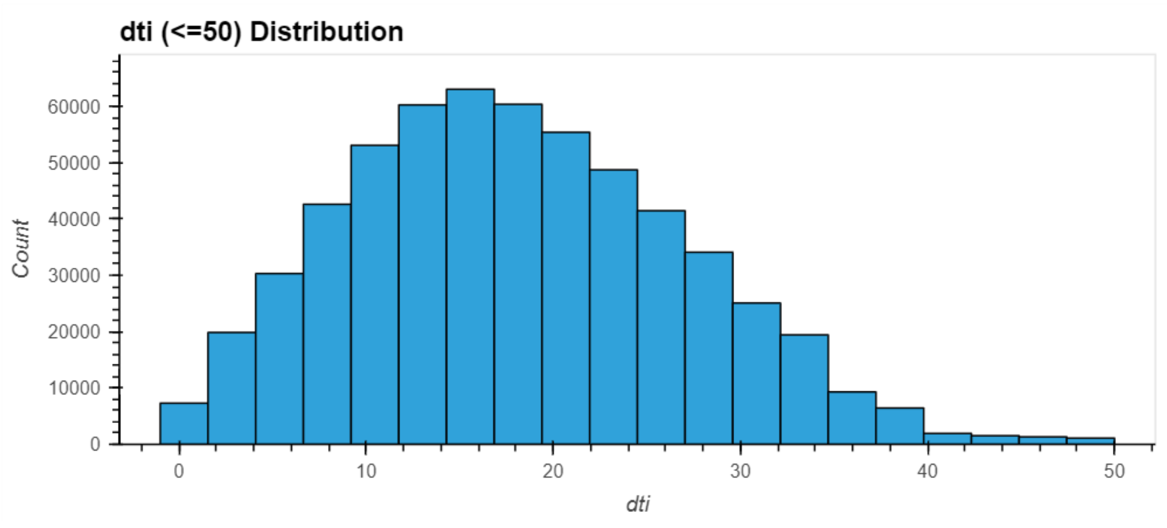


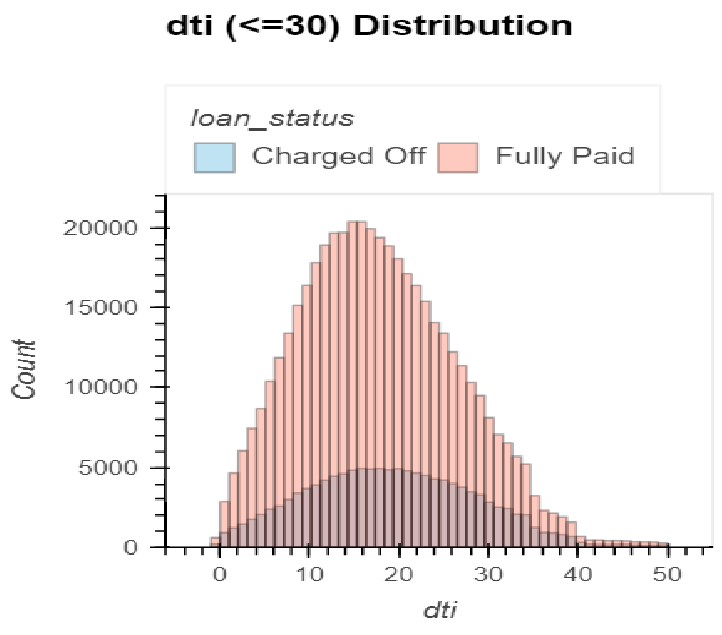


- Most of the loan ware given on the year of 2017 and 2018.
- After 2019 they didn't give many loans
- Charged off loans were also high in 2017 and 2018

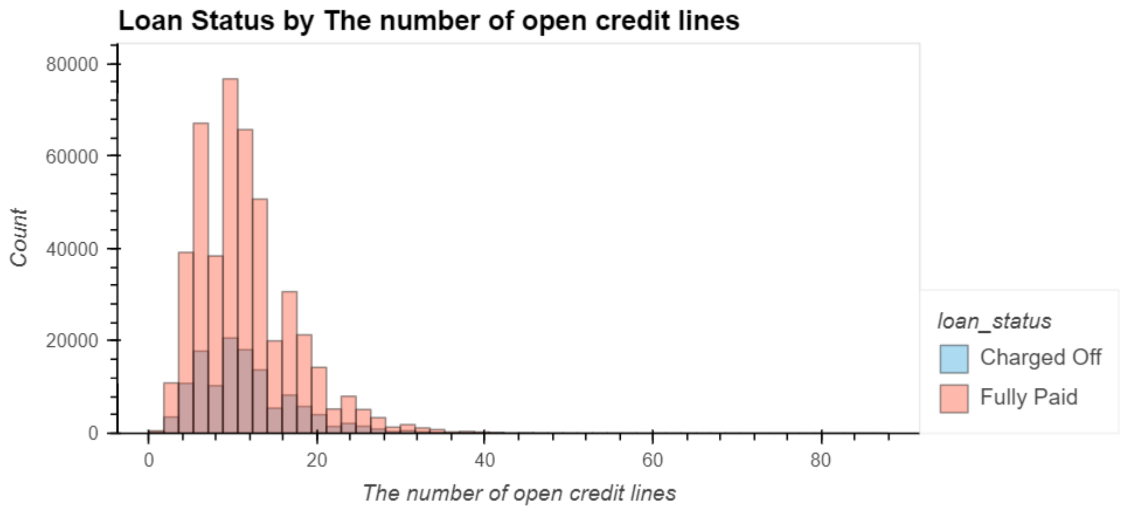
Dti, open account, revol bal, revol util, & total acc:

- dti: A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.
- open_acc: The number of open credit lines in the borrower's credit file.
- revol_bal: Total credit revolving balance
- revol_util: Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
- total_acc: The total number of credit lines currently in the borrower's credit file

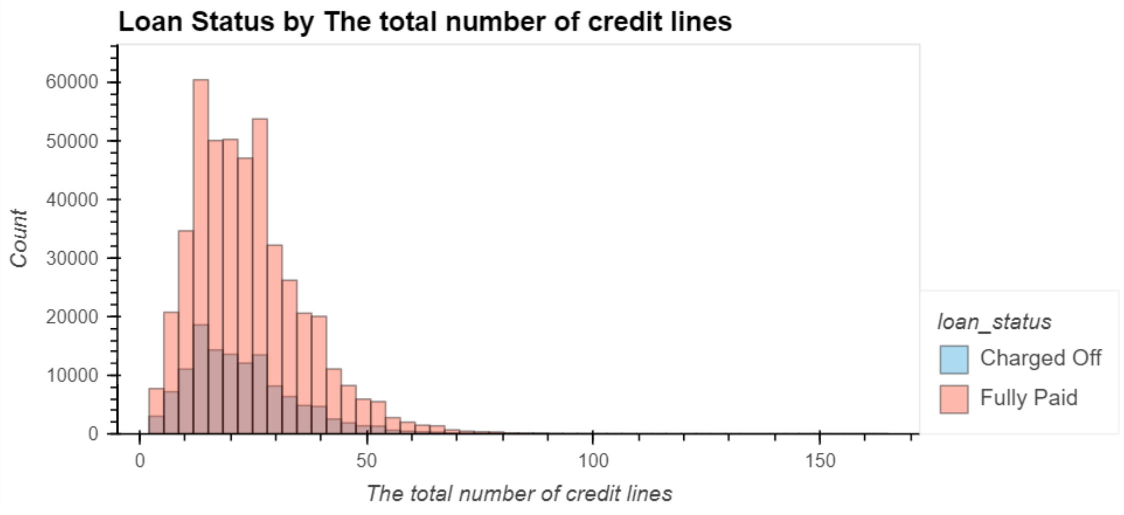




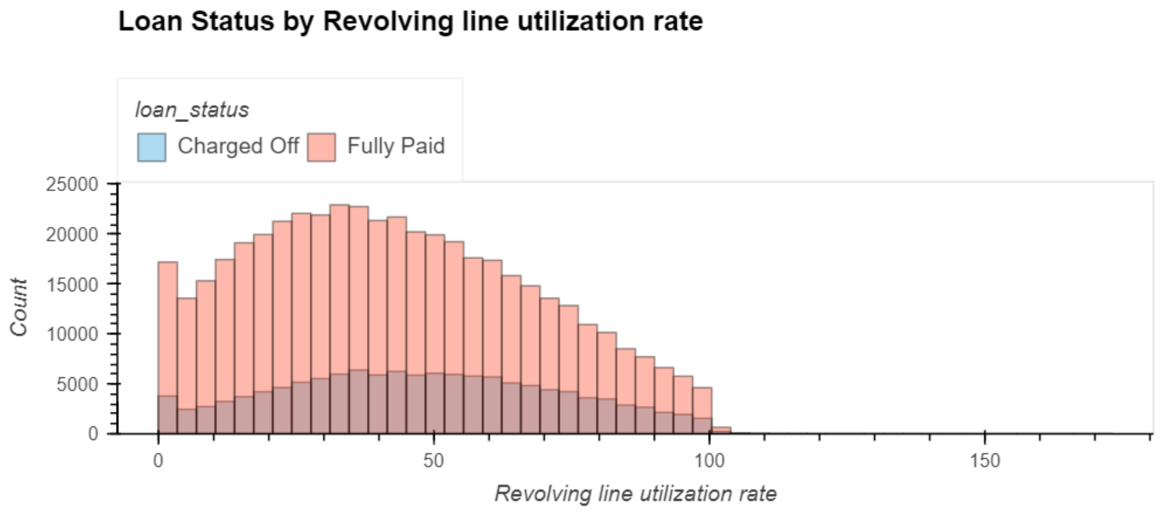
- Debt to income ratio is equally distributed for both charged off and fully paid



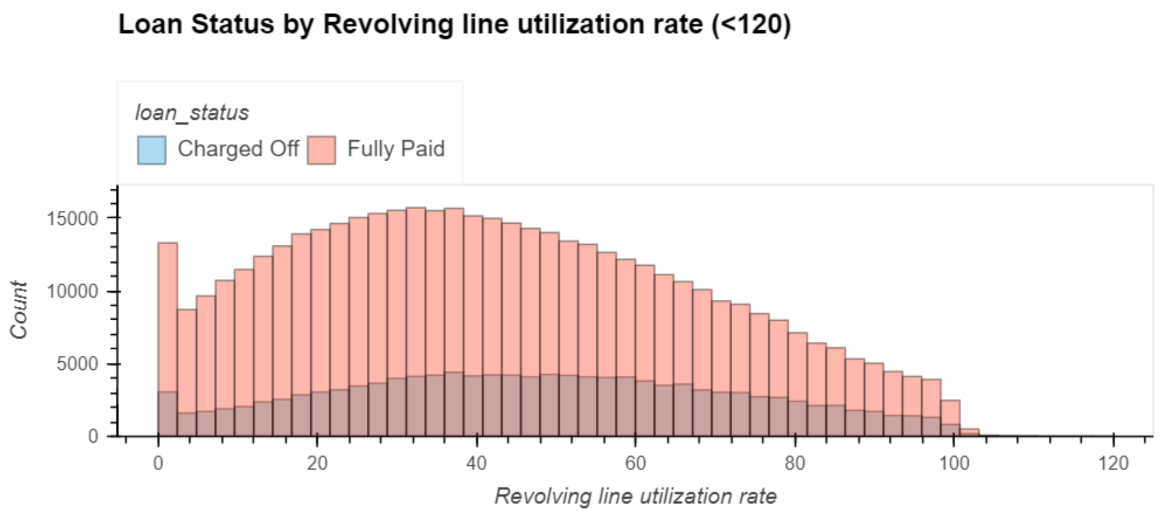
Loan Status by The number of open credit lines
Count
The number of open credit lines
loan_status
Charged Off
Fully Paid



Loan Status by The total number of credit lines
Count
The total number of credit lines
loan_status
Charged Off
Fully Paid

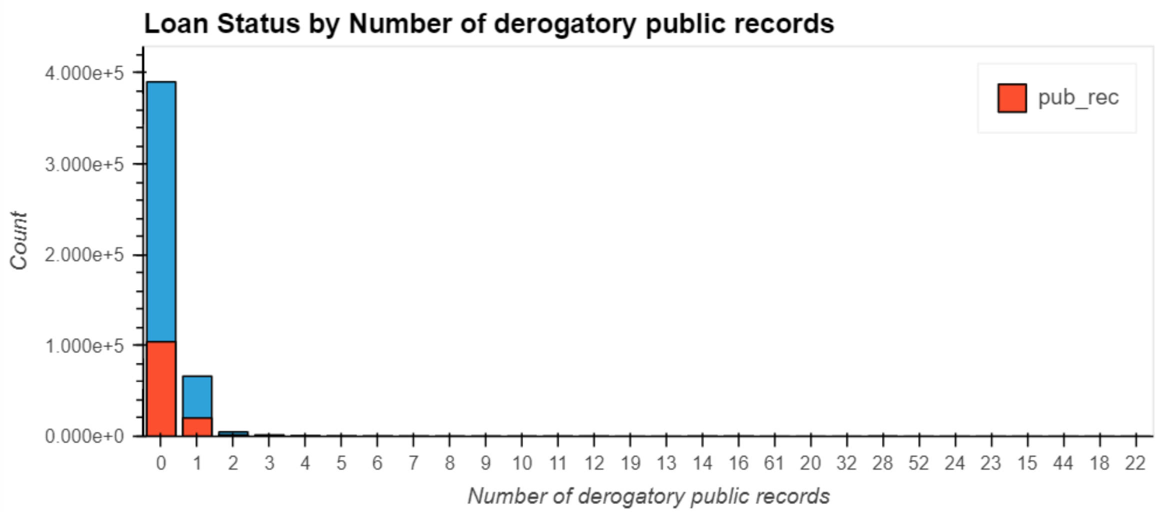


Loan Status by Revolving line utilization rate
loan_status
Charged Off
Fully Paid
Count
Revolving line utilization rate

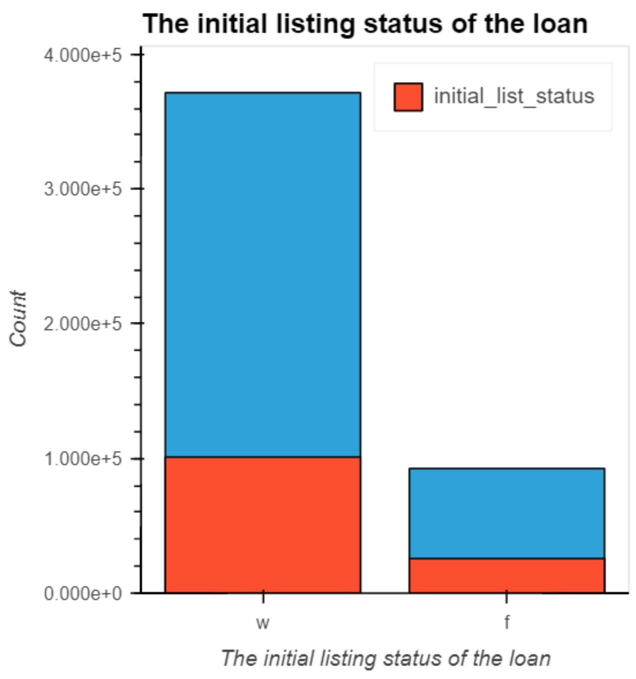


pub_rec, initial_list_status, application_type, mort_acc, & pub_rec_bankruptcies

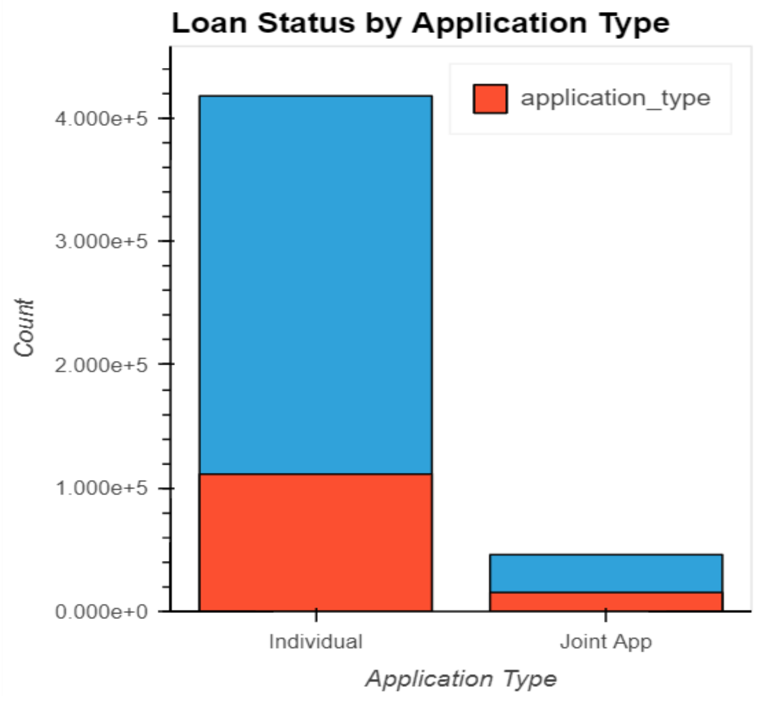
- **pub_rec**: Number of derogatory public records
- **initial_list_status**: The initial listing status of the loan. Possible values are – W, F
- **application_type**: Indicates whether the loan is an individual application or a joint application with two co-borrowers
- **mort_acc**: Number of mortgage accounts
- **pub_rec_bankruptcies**: Number of public record bankruptcies



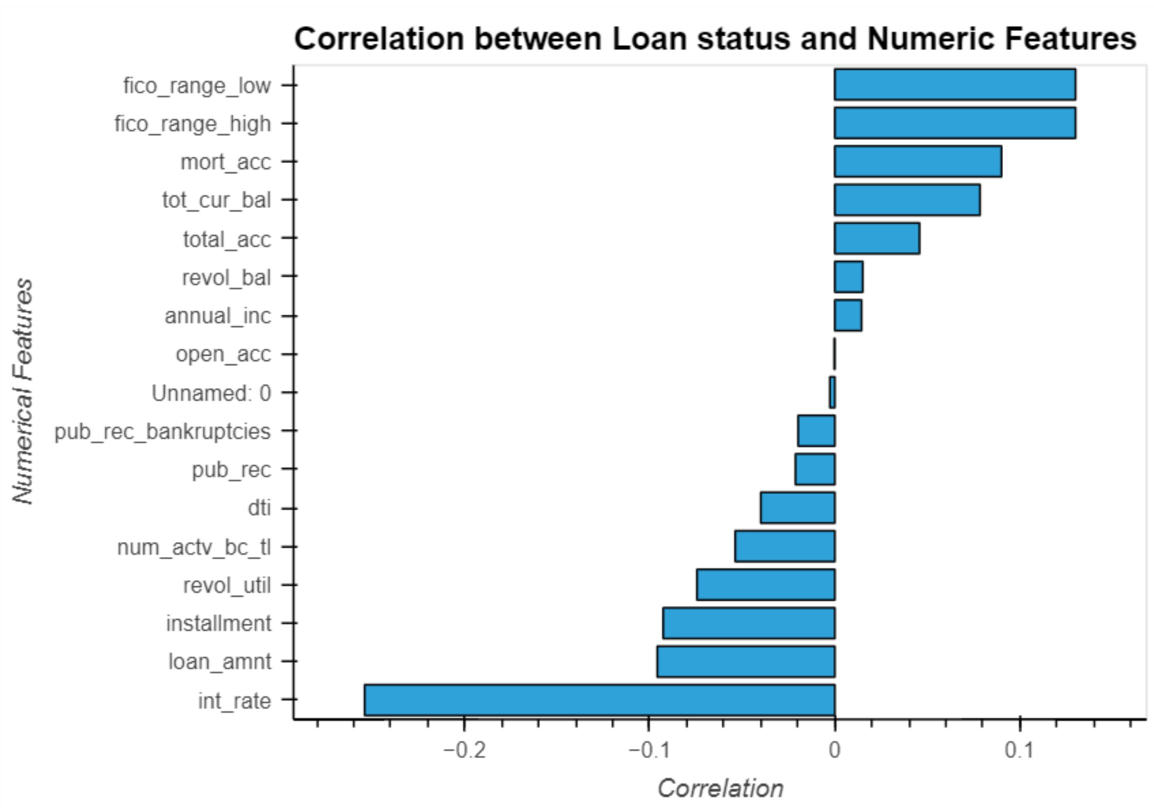
- It seems like data is having outliers



- Loan listed as w(whole) are having more charged off borrowers
- Very less number of loans were given to F(fractional)

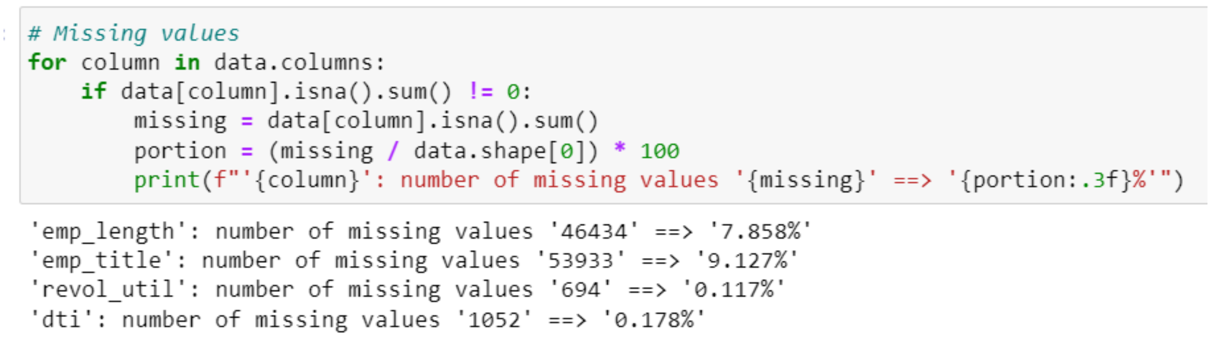


- Majority of loan application type was individual
- Most of the charged of category falls under individual application type
- Only a few Joint application type borrowers are there



- Loan status is positively correlated with fico_range high and low
- Loan status is negatively correlated with interest rate

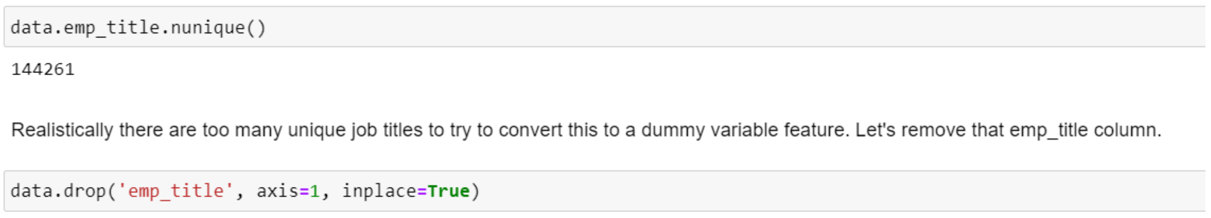
Filling missing value:



```
# Missing values
for column in data.columns:
    if data[column].isna().sum() != 0:
        missing = data[column].isna().sum()
        portion = (missing / data.shape[0]) * 100
        print(f"'{column}': number of missing values '{missing}' ==> '{portion:.3f}%'")
```

```
'emp_length': number of missing values '46434' ==> '7.858%'
'emp_title': number of missing values '53933' ==> '9.127%'
'revol_util': number of missing values '694' ==> '0.117%'
'dti': number of missing values '1052' ==> '0.178%'
```

Filling missing values of Employees titles



```
data.emp_title.nunique()
```

```
144261
```

Realistically there are too many unique job titles to try to convert this to a dummy variable feature. Let's remove that emp_title column.

```
data.drop('emp_title', axis=1, inplace=True)
```

Filling missing values of Employee length

```
data.emp_length.unique()
```

```
array(['10+ years', '3 years', '< 1 year', '5 years', '4 years', nan,
       '6 years', '2 years', '8 years', '7 years', '1 year', '9 years'],
      dtype=object)
```

```
for year in data.emp_length.unique():
    print(f"{year} years in this position:")
    print(f"{data[data.emp_length == year].loan_status.value_counts(normalize=True)}")
    print('==========================================')
```

```
10+ years years in this position:
1    0.808506
0    0.191494
Name: loan_status, dtype: float64
==========================================
3 years years in this position:
1    0.782635
0    0.217365
Name: loan_status, dtype: float64
==========================================
< 1 year years in this position:
1    0.762819
0    0.237181
Name: loan_status, dtype: float64
==========================================
5 years years in this position:
1    0.786767
0    0.213233
Name: loan_status, dtype: float64
==========================================
4 years years in this position:
1    0.781147
0    0.218853
Name: loan_status, dtype: float64
==========================================
nan years in this position:
Series([], Name: loan_status, dtype: float64)
==========================================
6 years years in this position:
1    0.797847
0    0.202153
```

```
==========================================
8 years years in this position:
1    0.792387
0    0.207613
Name: loan_status, dtype: float64
==========================================
7 years years in this position:
1    0.794626
0    0.205374
Name: loan_status, dtype: float64
==========================================
1 year years in this position:
1    0.775877
0    0.224123
Name: loan_status, dtype: float64
==========================================
9 years years in this position:
1    0.806211
0    0.193789
Name: loan_status, dtype: float64
==========================================
```

Charge off rates are extremely similar across all employment lengths. So we are going to drop the emp_length column.

```
data.drop('emp_length', axis=1, inplace=True)
```

num_actv_bc_tl

```
open_acc_avg = data.groupby(by='open_acc').mean().num_actv_bc_tl
```

```
def fill_num_actv_bc_tl(open_acc, num_actv_bc_tl):
    if np.isnan(num_actv_bc_tl):
        return open_acc_avg[open_acc].round()
    else:
        return num_actv_bc_tl
```

```
data['num_actv_bc_tl'] = data.apply(lambda x: fill_num_actv_bc_tl(x['open_acc'], x['num_actv_bc_tl']),
                                     axis=1)
```

Num_actv_bc_tl is correlated with open account, so we fill the missing value of num_actv_bc_tl with the help of open_acc.

revol_util & dti

These two features have missing data points, but they account for less than 0.5% of the total data. So we are going to remove the rows that are missing those values in those columns with dropna().

```
for column in data.columns:
    if data[column].isna().sum() != 0:
        missing = data[column].isna().sum()
        portion = (missing / data.shape[0]) * 100
        print(f"'{column}': number of missing values '{missing}' ==> '{portion:.3f}%'")
```

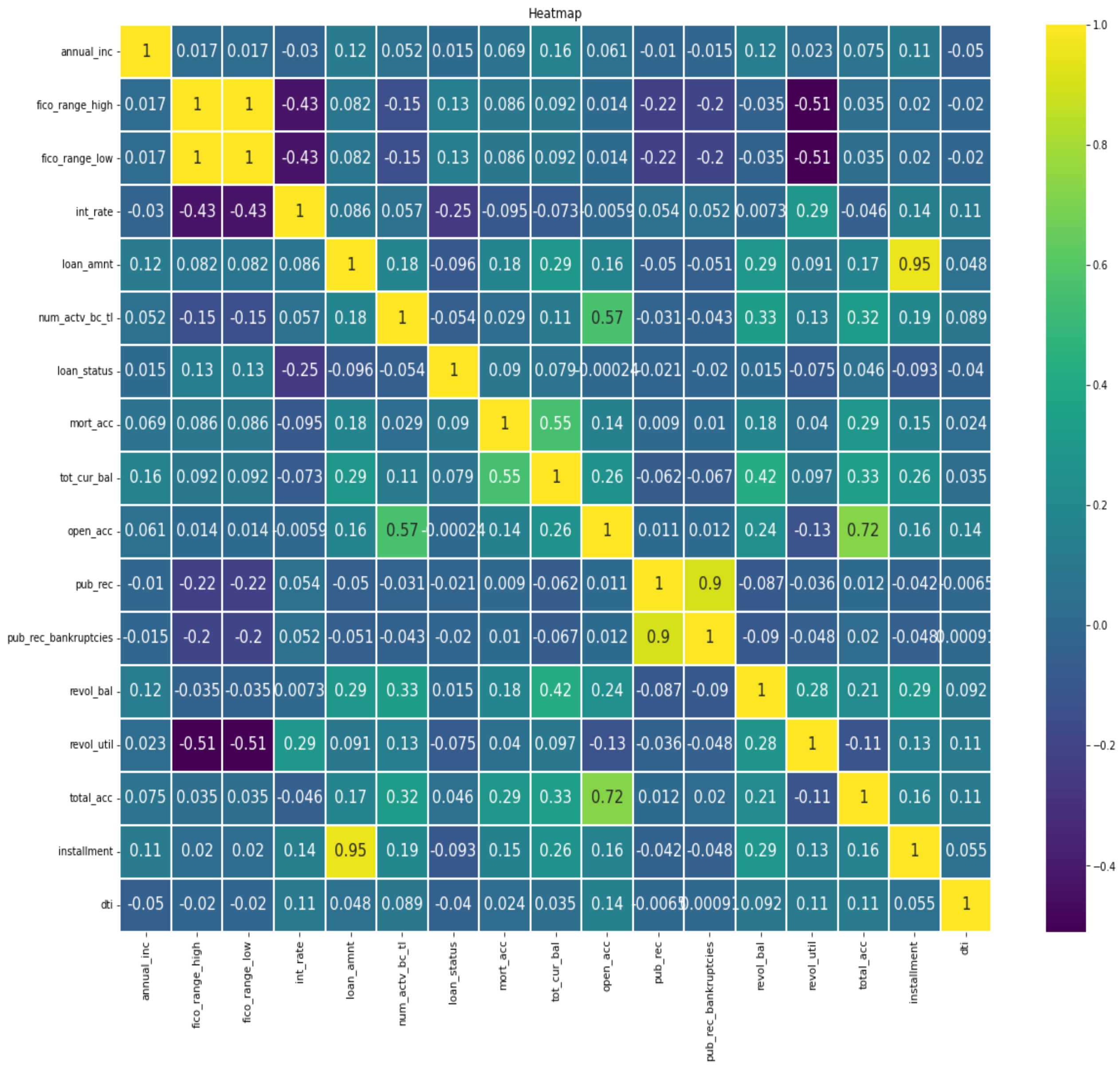
```
'revol_util': number of missing values '694' ==> '0.117%'
'dti': number of missing values '1052' ==> '0.178%'
```

```
data.dropna(inplace=True)
```

```
data.shape
```

```
(589155, 28)
```

Correlation:



- fico_range_high anf fico_range_low having high positive correlation so we can choose any one of those colums
- pub_rec and pub_rec_bankrupties are also having high positive correlatin so we can choose either one of those colums

```
data.drop('fico_range_low', axis=1, inplace=True)
```

```
data.drop('pub_rec_bankruptcies', axis=1, inplace=True)
```

Categorical Variable:

```
print([column for column in data.columns if data[column].dtype == object])
```

```
['grade', 'home_ownership', 'application_type', 'initial_list_status', 'purpose', 'verification_status']
```

```
data.term.unique()
```

```
array([' 36 months', ' 60 months'], dtype=object)
```

```
term_values = {' 36 months': 36, ' 60 months': 60}
data['term'] = data.term.map(term_values)
```

addr_state

```
data['addr_state'].unique()
```

```
array(['CA', 'OR', 'NY', 'TX', 'WI', 'CT', 'FL', 'AZ', 'SC', 'WY', 'WA',
       'VA', 'NC', 'NV', 'NM', 'IL', 'MI', 'MN', 'PA', 'KY', 'GA', 'NJ',
       'HI', 'MD', 'AL', 'MA', 'MO', 'NE', 'OH', 'CO', 'KS', 'IN', 'AR',
       'LA', 'NH', 'OK', 'RI', 'TN', 'DE', 'ME', 'UT', 'MS', 'DC', 'VT',
       'AK', 'ID', 'ND', 'MT', 'SD', 'WV'], dtype=object)
```

```
data.drop('addr_state', axis=1, inplace=True)
```

grade & sub_grade

We know that grade is just a sub feature of sub_grade, So we are goinig to drop sub_grade.

```
data.drop('sub_grade', axis=1, inplace=True)
```

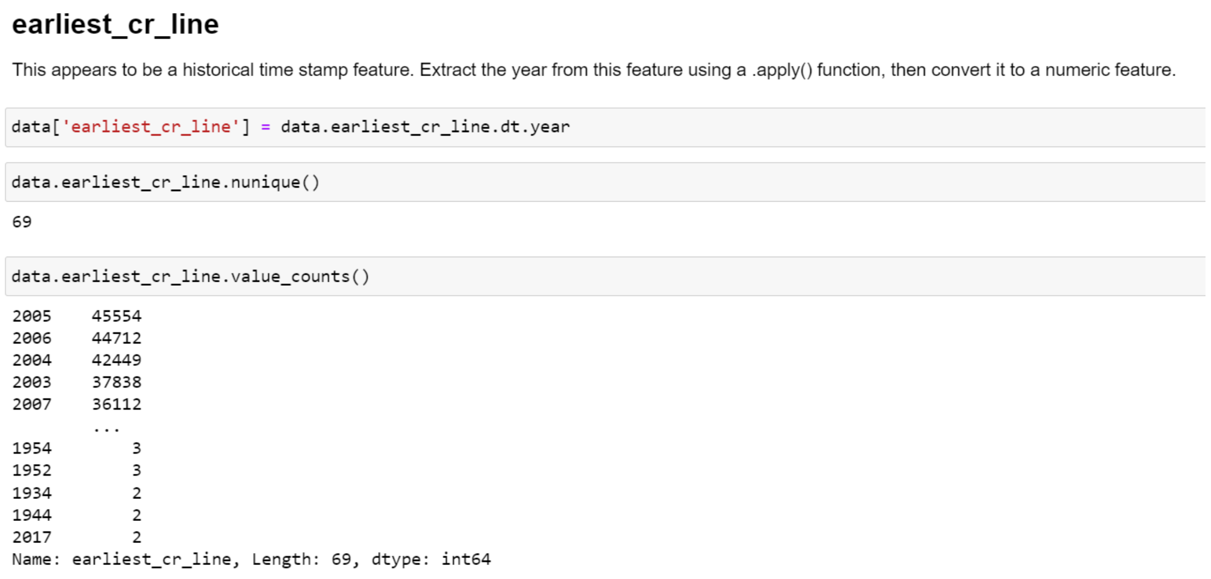
dummies

```
dummies = ['grade', 'verification_status', 'purpose', 'initial_list_status',
           'application_type', 'home_ownership']
data = pd.get_dummies(data, columns=dummies, drop_first=True)
```

issue_d

we wouldn't know beforehand whether or not a loan would be issued when using our model, so in theory we wouldn't have an issue_date, drop this feature.

```
data.drop('issue_d', axis=1, inplace=True)
```



earliest_cr_line

This appears to be a historical time stamp feature. Extract the year from this feature using a .apply() function, then convert it to a numeric feature.

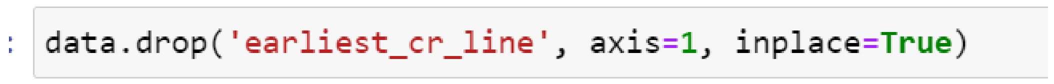
```
data['earliest_cr_line'] = data.earliest_cr_line.dt.year
```

```
data.earliest_cr_line.nunique()
```

```
69
```

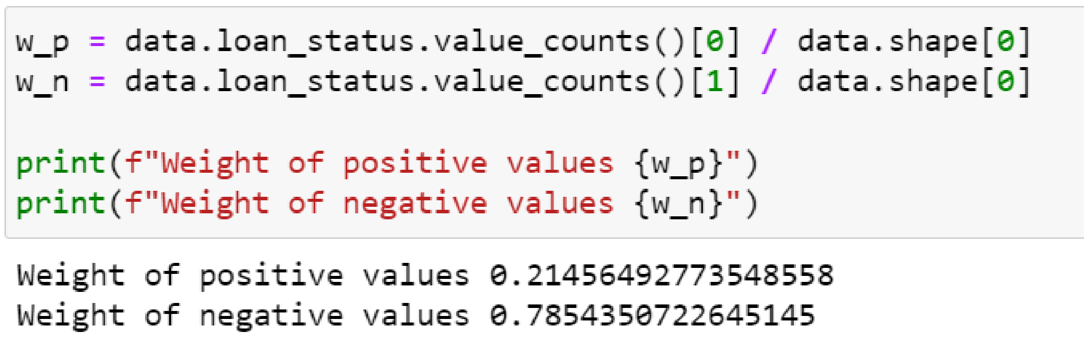
```
data.earliest_cr_line.value_counts()
```

```
2005    45554
2006    44712
2004    42449
2003    37838
2007    36112
        ...
1954        3
1952        3
1934        2
1944        2
2017        2
Name: earliest_cr_line, Length: 69, dtype: int64
```



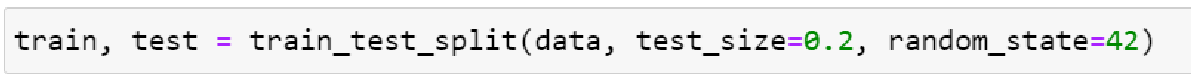
```
data.drop('earliest_cr_line', axis=1, inplace=True)
```

Train Test Split:



```
w_p = data.loan_status.value_counts()[0] / data.shape[0]
w_n = data.loan_status.value_counts()[1] / data.shape[0]

print(f"Weight of positive values {w_p}")
print(f"Weight of negative values {w_n}")
```

```
Weight of positive values 0.21456492773548558
Weight of negative values 0.7854350722645145
```



```
train, test = train_test_split(data, test_size=0.2, random_state=42)
```

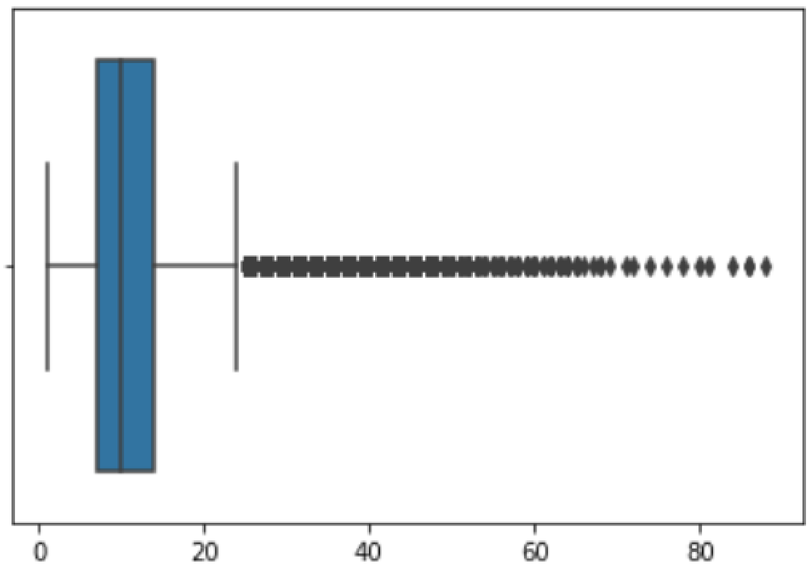
Removing Outliers:

We are using z score method to remove outliers

```
sns.boxplot(data['open_acc'])
```

```
C:\Users\ssuri\anaconda3\lib\site-packages\seaborn\_decorators
rg: x. From version 0.12, the only valid positional argument w
yword will result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='open_acc'>
```



```
def remove_outliers(col):
    low = train[col].mean()-3*train[col].std()
    upp = train[col].mean()+3*train[col].std()
    return train.loc[(train[col]<upp)&(train[col]>low)]
```

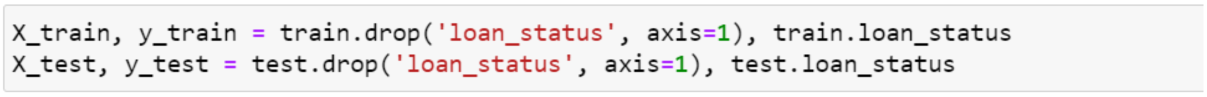
```
remove = ['annual_inc', 'fico_range_high', 'fico_range_low', 'int_rate',
          'loan_amnt', 'num_actv_bc_tl',  'mort_acc', 'tot_cur_bal',
          'open_acc','total_acc','revol_util','dti','pub_rec','pub_rec_bankruptcies']
```

```
for col in remove:
    train=remove_outliers(col)
```

```
train.shape
```

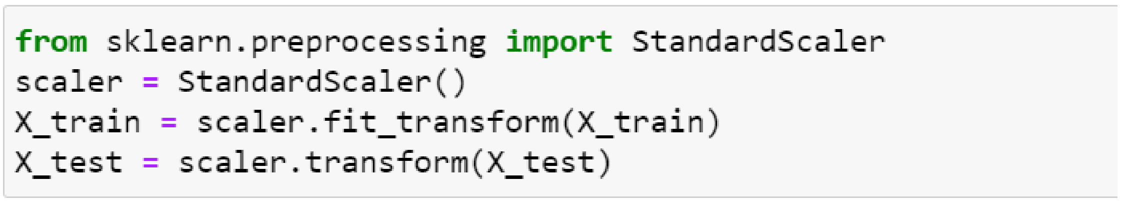
```
(434467, 121)
```

Splitting X and y:



```
X_train, y_train = train.drop('loan_status', axis=1), train.loan_status
X_test, y_test = test.drop('loan_status', axis=1), test.loan_status
```

Normalizing data:



```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

StandardScaler comes into play when the characteristics of the input dataset differ greatly between their ranges, or simply when they are measured in different units of measure.

The idea behind the StandardScaler is that variables that are measured at different scales do not contribute equally to the fit of the model and the learning function of the model and could end up creating a bias.

So, to deal with this potential problem, we need to standardize the data ($\mu = 0$, $\sigma = 1$) that is typically used before we integrate it into the machine learning model. StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes mean = 0 and scales the data to unit variance.

Chapter – VI

Model Development

For this project we used the following models to predict the loan status

- Logistic Regression
- Random Forest Classifier
- XGBoost Classifier
- KNN Classifier

Understanding classification models

Classification algorithms, or classifiers as they're also known, fall into the supervised learning branch of machine learning. As the name suggests, these predictive models are designed to determine the class to which a given subject belongs. Since they use supervised learning, they require labeled training data that includes a column containing their class.

The basic classification modeling process involves obtaining a dataset, creating features of independent variables, and using them to predict a dependent variable or target class. Most classification datasets require some preparation before they can be used by classifiers, and also usually require the creation of additional features through a process called feature engineering.

After importing the data into the model, it is trained on a subset of the full dataset. The model will learn to identify which of the independent variables or features is correlated with the target variable or class, and will iterate over the data, progressively becoming more accurate at making predictions. Once trained, the classification model can be evaluated to assess its accuracy and used to make predictions on unlabelled data.

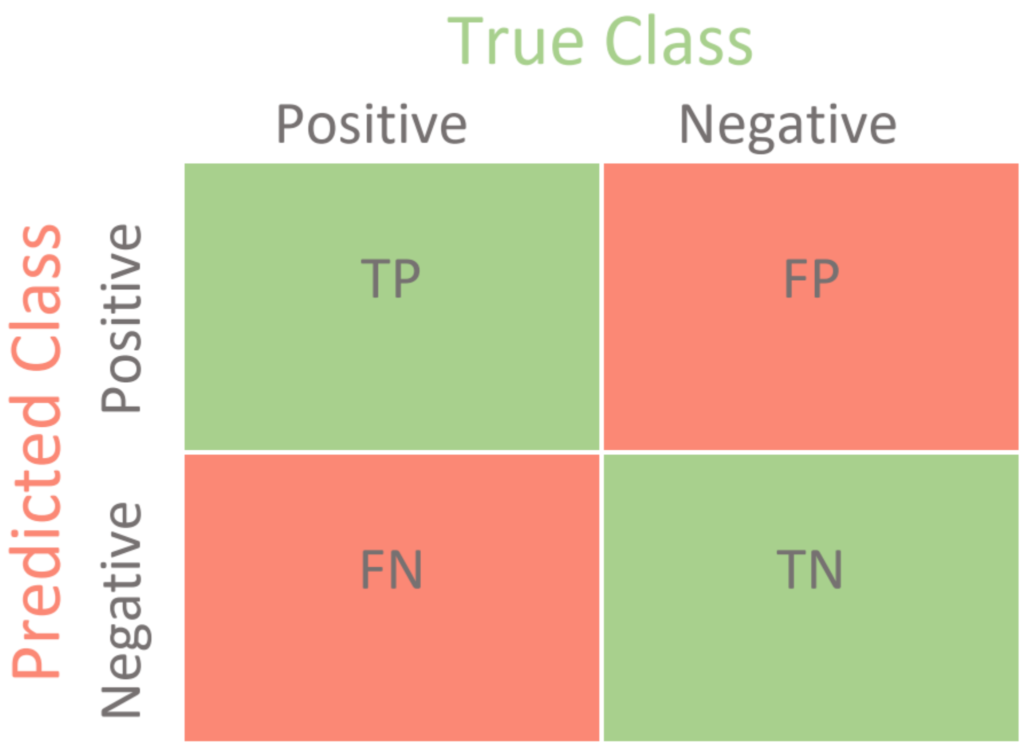
Accuracy Metrics

Then we compared the results of these models by using the following metrics

- Accuracy Score
- Confusion Matrix
- Classification Report
- ROC curve
- AUC score

Confusion Matrix:

Confusion matrices represent counts from predicted and actual values.



- "TN" stands for True Negative which shows the number of negative examples classified accurately.
- "TP" stands for True Positive which indicates the number of positive examples classified accurately.
- "FP" shows False Positive value, i.e., the number of actual negative examples classified as positive
- "FN" means a False Negative value which is the number of actual positive examples classified as negative.

One of the most commonly used metrics while performing classification is accuracy. The accuracy of a model (through a confusion matrix) is calculated using the given formula below

$$\text{Accuracy} = \frac{\text{TN+TP}}{\text{TN+FP+FN+TP}}$$

Classification Report:

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report.

Precision – What percent of your predictions were correct?

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives.

Precision – Accuracy of positive predictions.

Precision = TP/(TP + FP)

Recall – What percent of the positive cases did you catch?

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

Recall: Fraction of positives that were correctly identified.

Recall = TP/(TP+FN)

F1 score – What percent of positive predictions were correct?

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Roc Curve:

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR) is defined as follows:

$$FPR = \frac{FP}{FP + TN}$$

AUC Score:

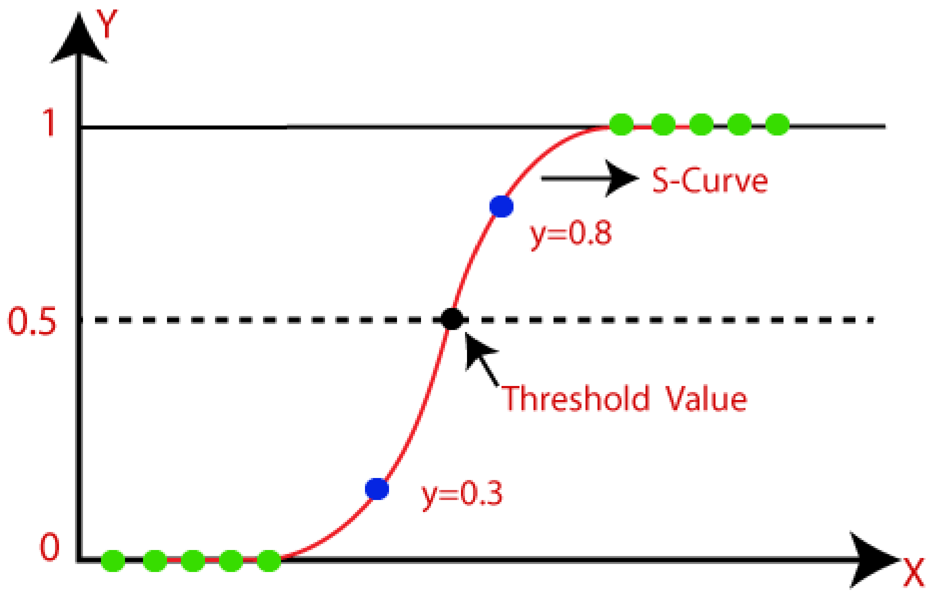
The area under the curve is one of the good ways to estimate the accuracy of the model. An excellent model poses an AUC near to the 1 which tells that it has a good measure of separability. A poor model will have an AUC near 0 which describes that it has the worst measure of separability. In fact, it means it is reciprocating the result and predicting 0s as 1s and 1s as 0s. When an AUC is 0.5, it means the model has no class separation capacity present whatsoever.

Logistic Regression:

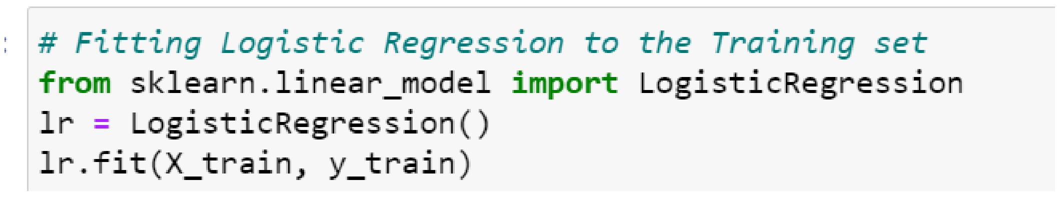
This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formula:

$$\mathit{Logistic\ function} = \frac{1}{1+e^{-x}}$$

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

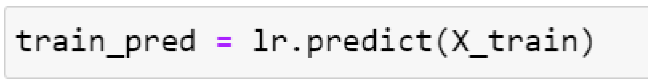


Building logistic regression model



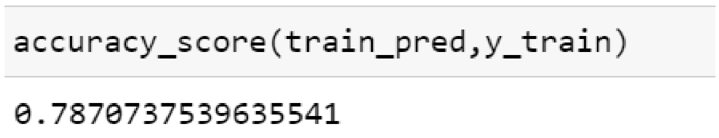
```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

Predicting X_test value:



```
train_pred = lr.predict(X_train)
```

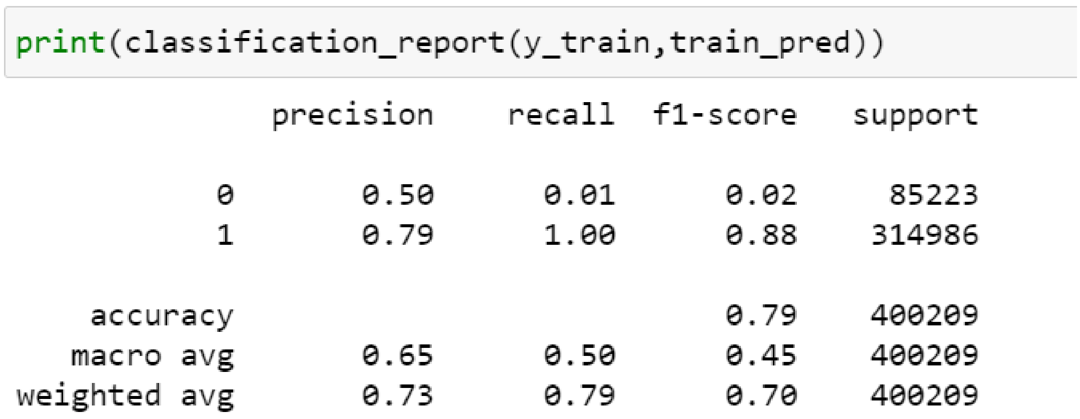
Accuracy score for train data



```
accuracy_score(train_pred,y_train)

0.7870737539635541
```

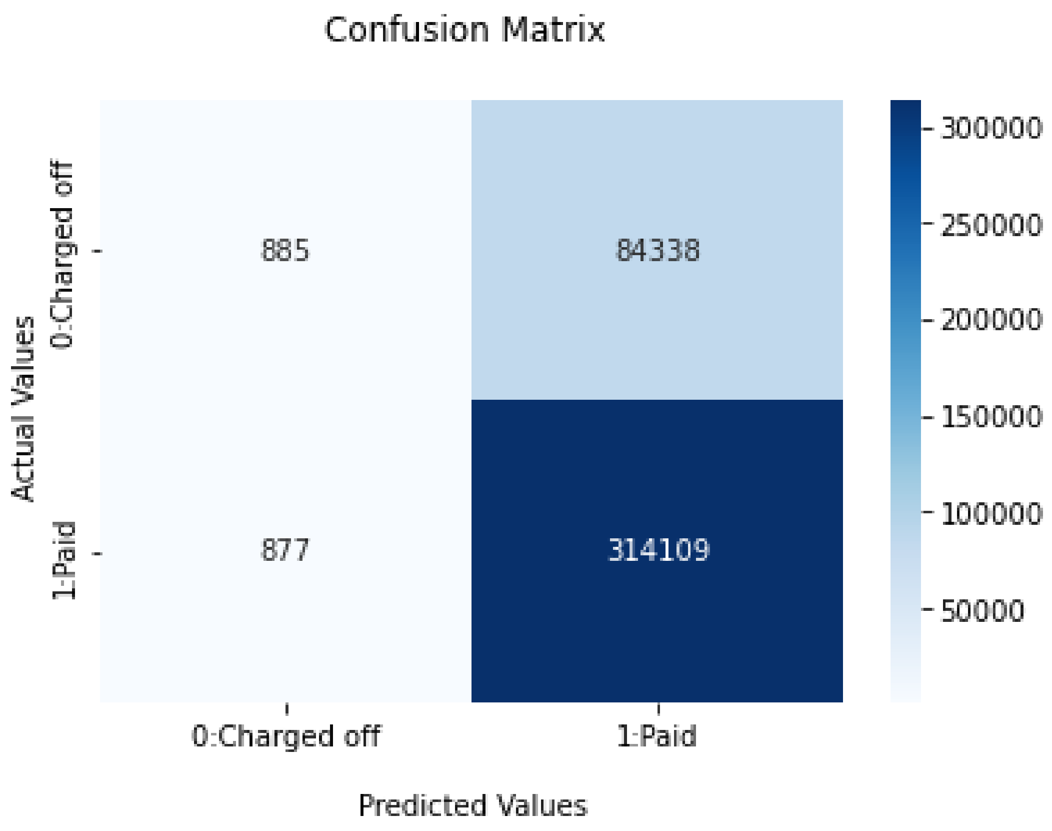
Our model is having 78 percent accuracy when we predicting train data.

Classification Report for train dataset

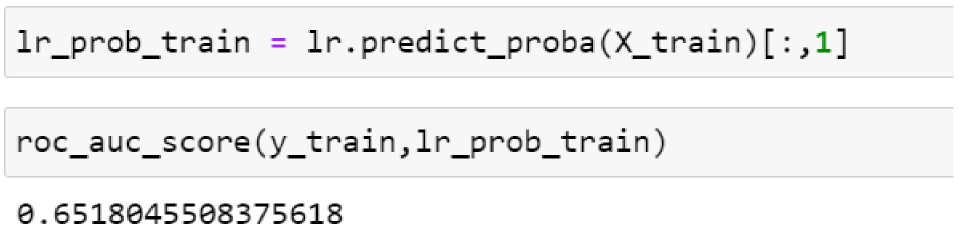


```
print(classification_report(y_train,train_pred))

              precision    recall  f1-score   support

           0       0.50      0.01      0.02     85223
           1       0.79      1.00      0.88    314986

    accuracy                           0.79    400209
   macro avg       0.65      0.50      0.45    400209
weighted avg       0.73      0.79      0.70    400209
```

Confusion matrix for train dataset



AUC Score:



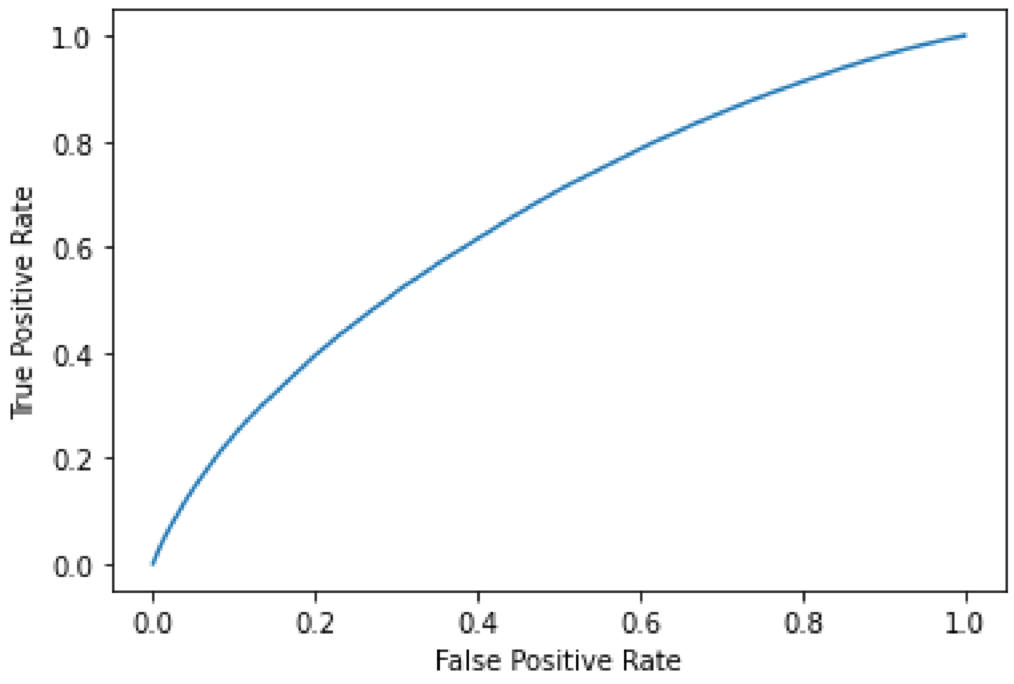
```
lr_prob_train = lr.predict_proba(X_train)[:,1]
```

```
roc_auc_score(y_train,lr_prob_train)
```

```
0.6518045508375618
```

AUC score for training dataset is 0.65

ROC Curve



Accuracy Score for test dataset

```
test_pred = lr.predict(X_test)
```

```
accuracy_score(y_test,test_pred)
```

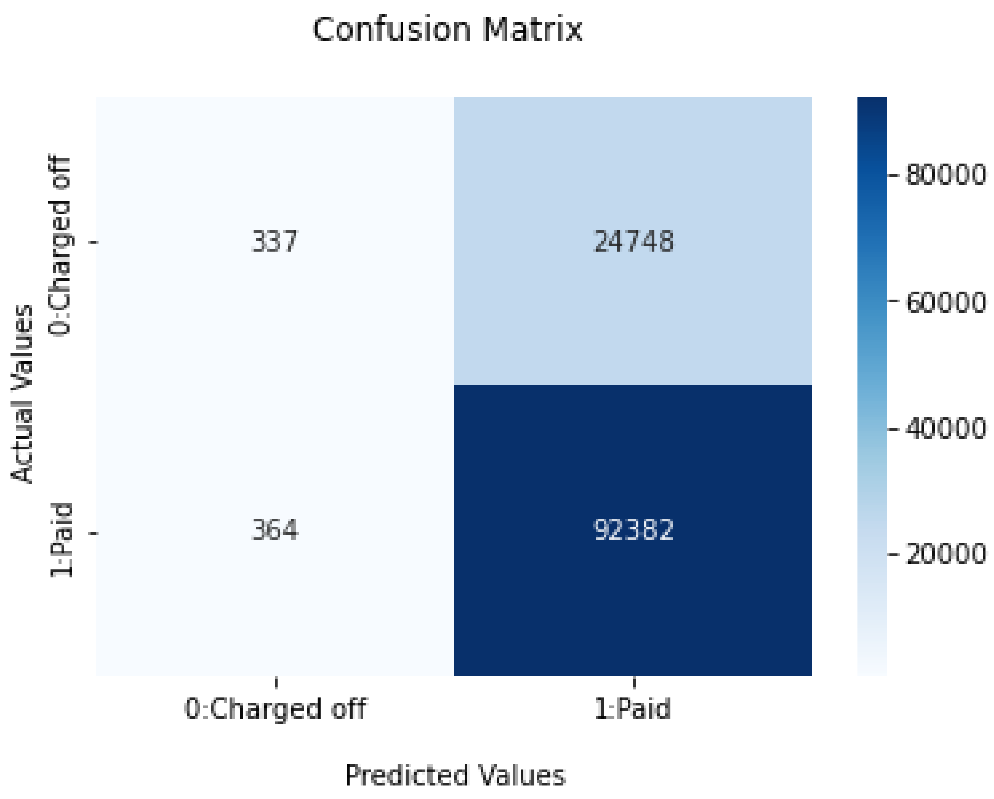
```
0.7868812112262478
```

Classification Report

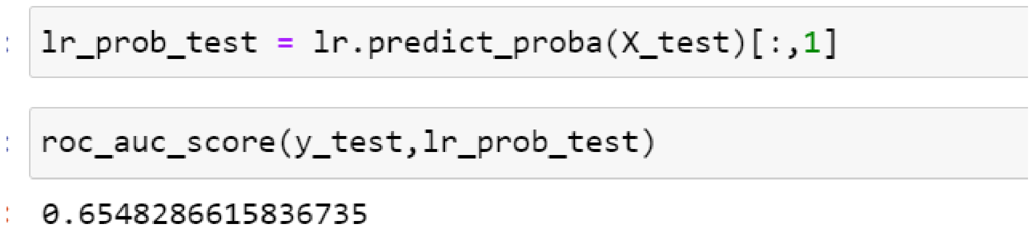
```
print(classification_report(y_test,test_pred))
```

```
              precision    recall  f1-score   support

           0       0.48      0.01      0.03     25085
           1       0.79      1.00      0.88     92746

    accuracy                           0.79    117831
   macro avg       0.63      0.50      0.45    117831
weighted avg       0.72      0.79      0.70    117831
```

Confusion Matrix



AUC Score for test dataset

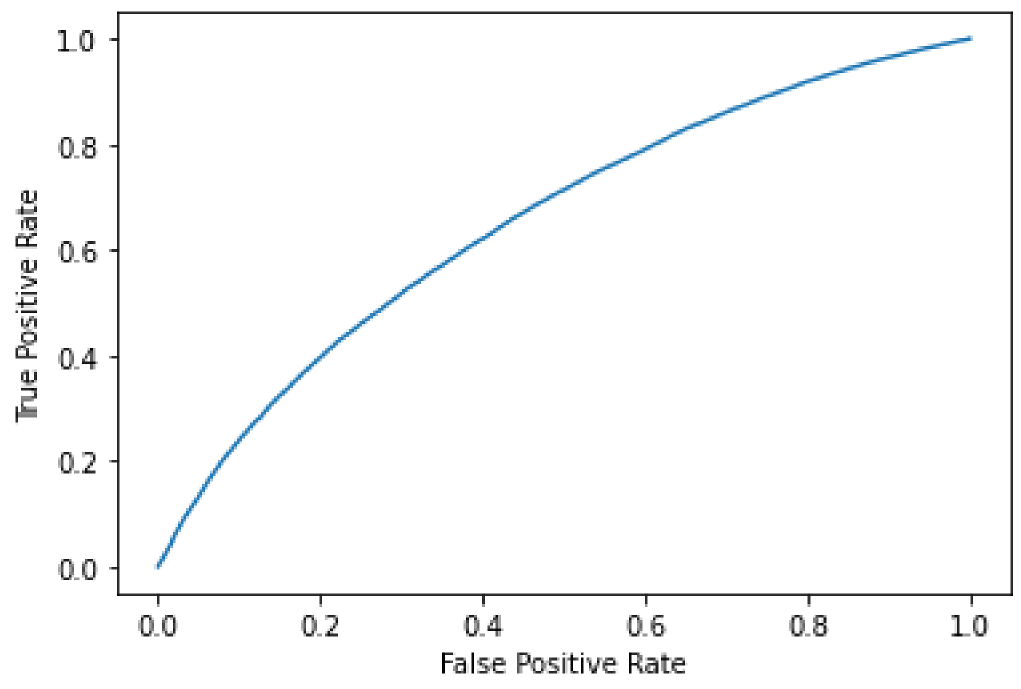


```
lr_prob_test = lr.predict_proba(X_test)[:,1]
```

```
roc_auc_score(y_test,lr_prob_test)
```

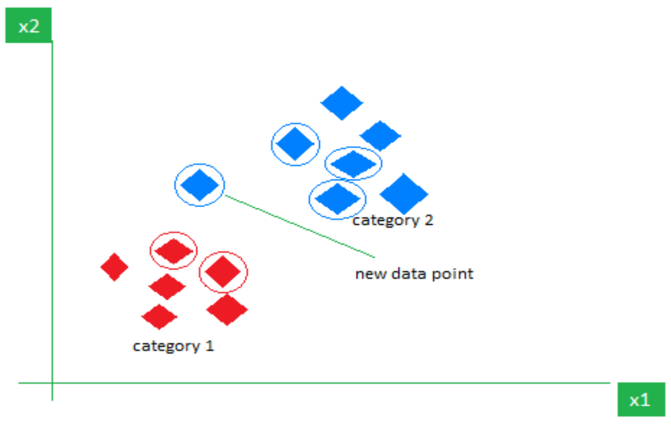
```
0.6548286615836735
```

ROC Curve for test dataset



KNN Classifier:
In the KNN algorithm, K specifies the number of neighbors and its algorithm is as follows:

- Choose the number K of neighbor.
- Take the K Nearest Neighbor of unknown data point according to distance.
- Among the K-neighbors, Count the number of data points in each category.
- Assign the new data point to a category, where you counted the most neighbors.
- For the Nearest Neighbor classifier, the distance between two points is expressed in the form of Euclidean Distance.



So, when a new data point enters, out of 5 neighbors, 3 are Blue and 2 are Red. We assign the new data point to the category with most neighbors i.e Blue.

Building KNN Classifier model for our dataset

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier()
```

Predicting training and test data

```
train_pred = knn.predict(X_train)
```

```
test_pred = knn.predict(X_test)
```

```
print(accuracy_score(y_train, train_pred))
print(accuracy_score(y_test, test_pred))
```

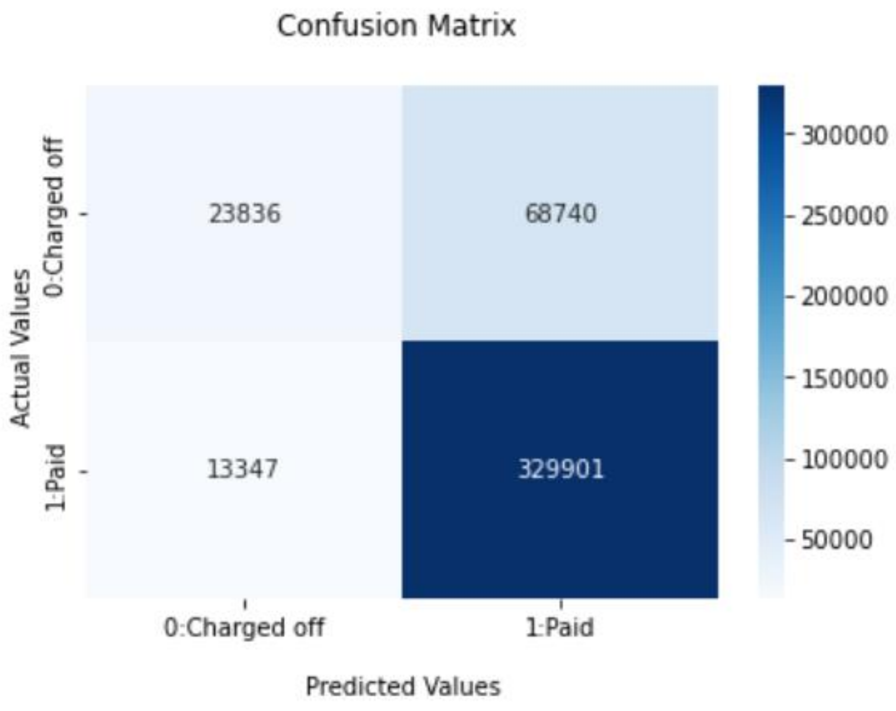
```
0.8116510334446932
0.7516527908614881
```

Classification report

```
print(classification_report(y_train,train_pred))
```

```
              precision    recall  f1-score   support

           0       0.64      0.26      0.37     92576
           1       0.83      0.96      0.89    343248

    accuracy                           0.81    435824
   macro avg       0.73      0.61      0.63    435824
weighted avg       0.79      0.81      0.78    435824
```

Confusion Matrix



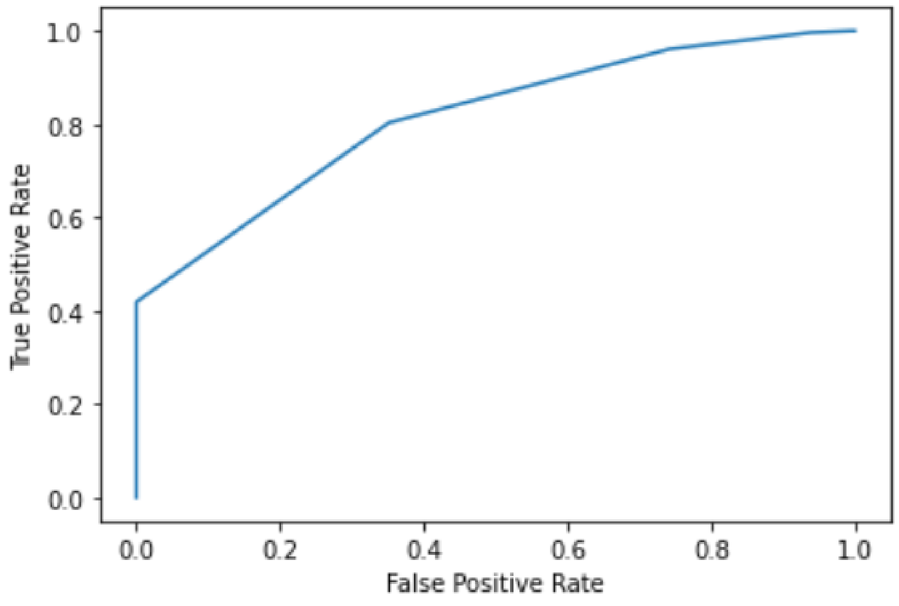
AUC Score

```
knn_prob_train = knn.predict_proba(X_train)[:,1]
```

```
roc_auc_score(y_train,knn_prob_train)
```

```
0.8129848554901452
```

ROC Curve

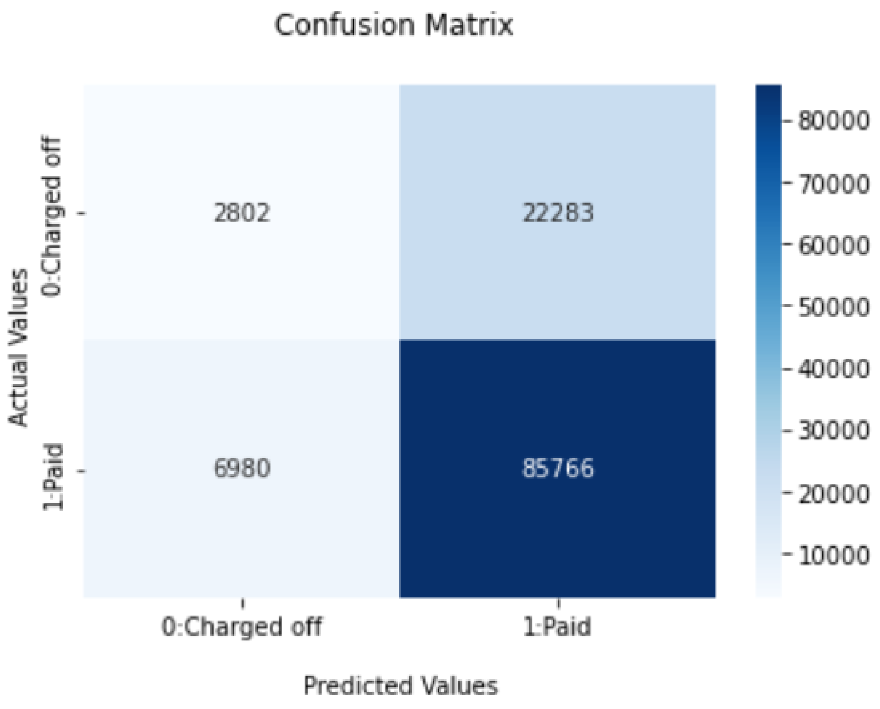


Classification report for test data set

```
print(classification_report(y_test,test_pred))
```

```
              precision    recall  f1-score   support

           0       0.29      0.11      0.16     25085
           1       0.79      0.92      0.85     92746

    accuracy                           0.75    117831
   macro avg       0.54      0.52      0.51    117831
weighted avg       0.69      0.75      0.71    117831
```

Confusion Matrix



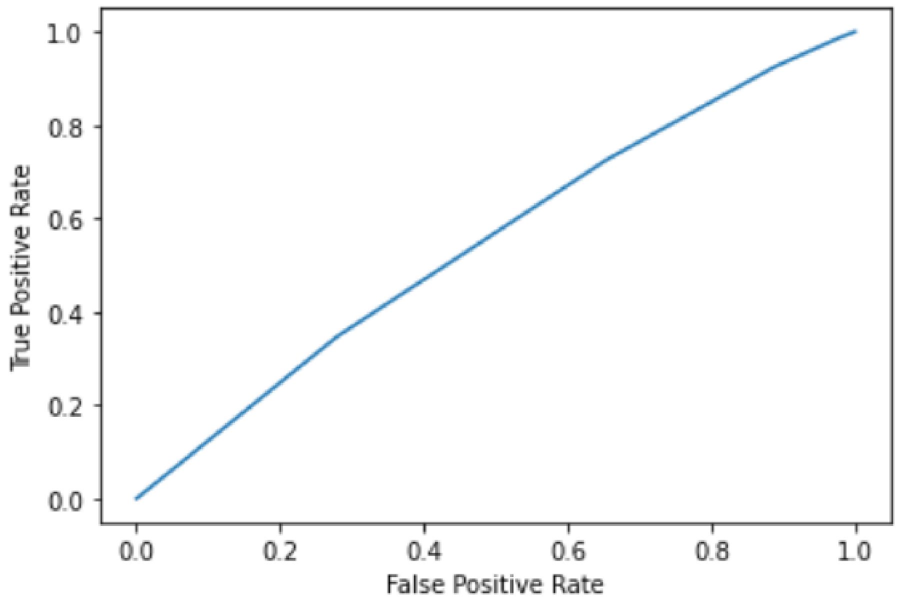
AUC Score for test dataset

```
knn_prob_test = knn.predict_proba(X_test)[:,1]
```

```
roc_auc_score(y_test,knn_prob_test)
```

```
0.5502889681261873
```

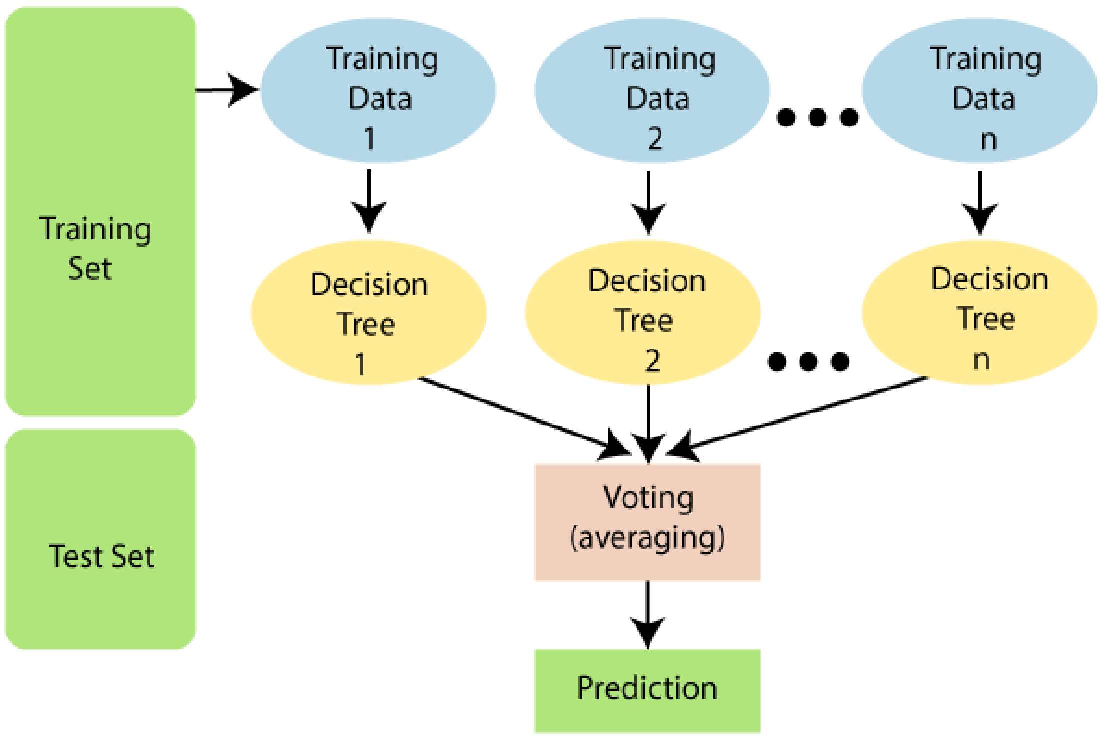
ROC Curve



Random Forest:

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

1. Select random K data points from the training set.

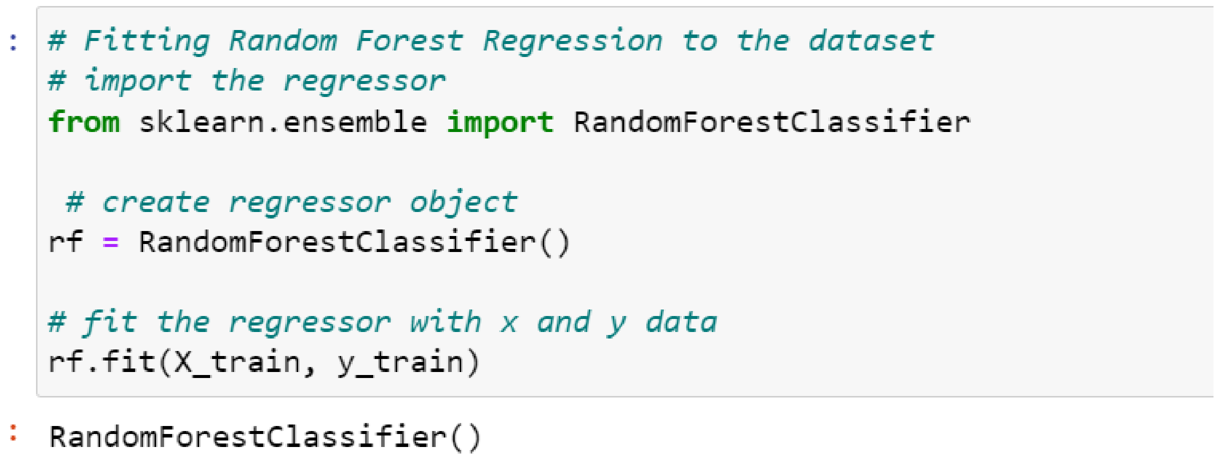
2. Build the decision trees associated with the selected data points (Subsets).

3. Choose the number N for decision trees that you want to build.

4. Repeat Step 1 & 2.

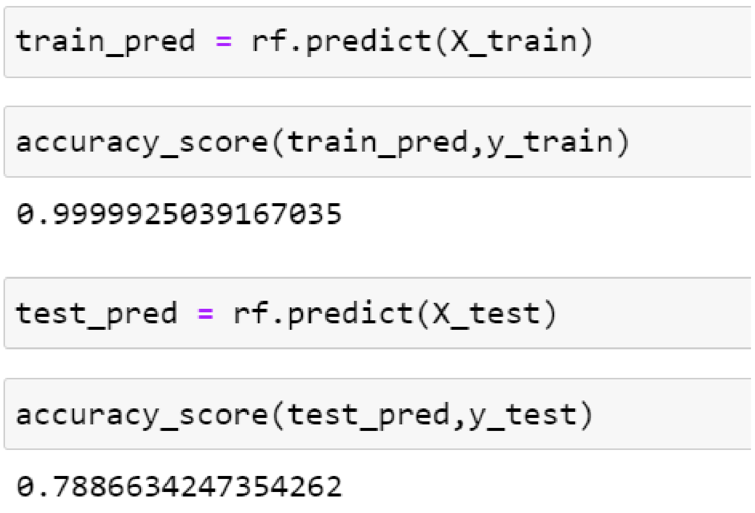
5. For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Create random forest classifier model for out dataset



```
: # Fitting Random Forest Regression to the dataset
  # import the regressor
  from sklearn.ensemble import RandomForestClassifier

   # create regressor object
  rf = RandomForestClassifier()

  # fit the regressor with x and y data
  rf.fit(X_train, y_train)

: RandomForestClassifier()
```

Predicting train and test set accuracy



```
train_pred = rf.predict(X_train)
```

```
accuracy_score(train_pred,y_train)
```

```
0.9999925039167035
```

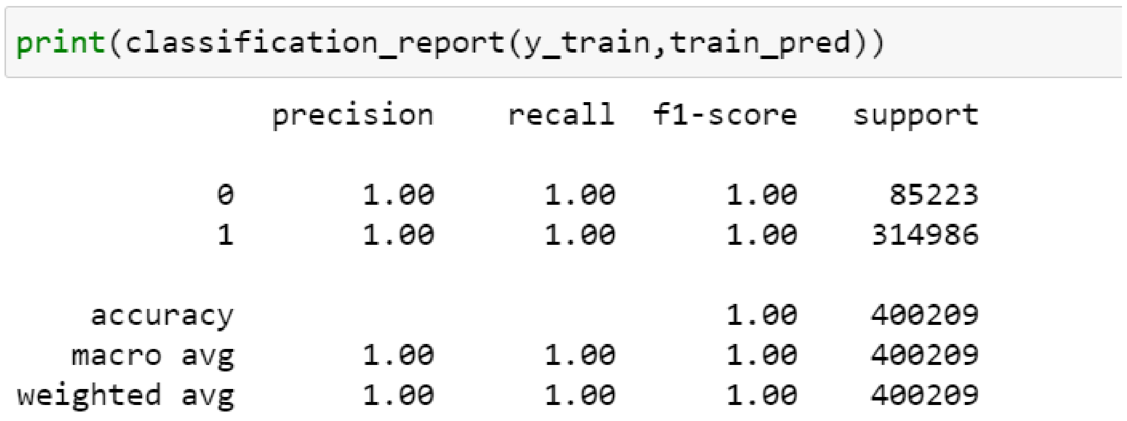
```
test_pred = rf.predict(X_test)
```

```
accuracy_score(test_pred,y_test)
```

```
0.7886634247354262
```

Accuracy score for test dataset is 78 percent and for training data accuracy is 99 percent

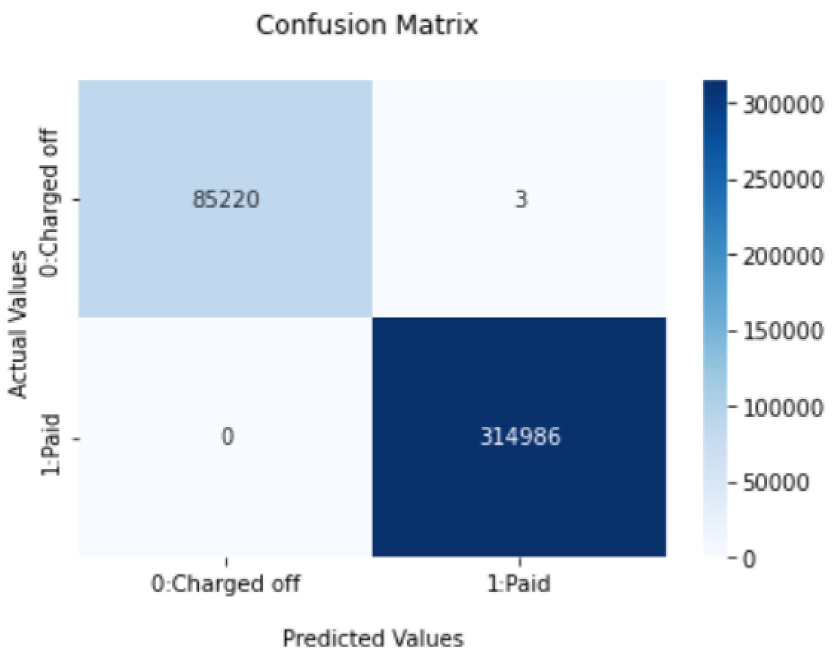
Classification report for training set



```
print(classification_report(y_train,train_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85223
           1       1.00      1.00      1.00    314986

    accuracy                           1.00    400209
   macro avg       1.00      1.00      1.00    400209
weighted avg       1.00      1.00      1.00    400209
```

Confusion matrix for train set



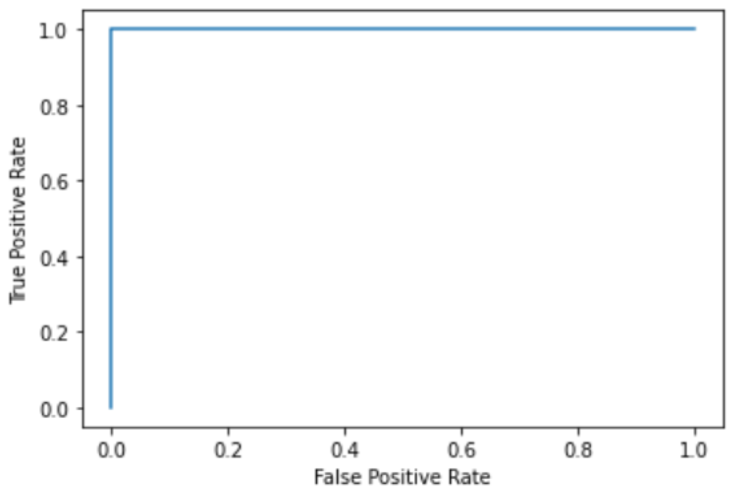
AUC score for training dataset

```
rf_prob_train = rf.predict_proba(X_train)[:,1]
```

```
roc_auc_score(y_train,rf_prob_train)
```

```
1.0
```

ROC curve for training dataset

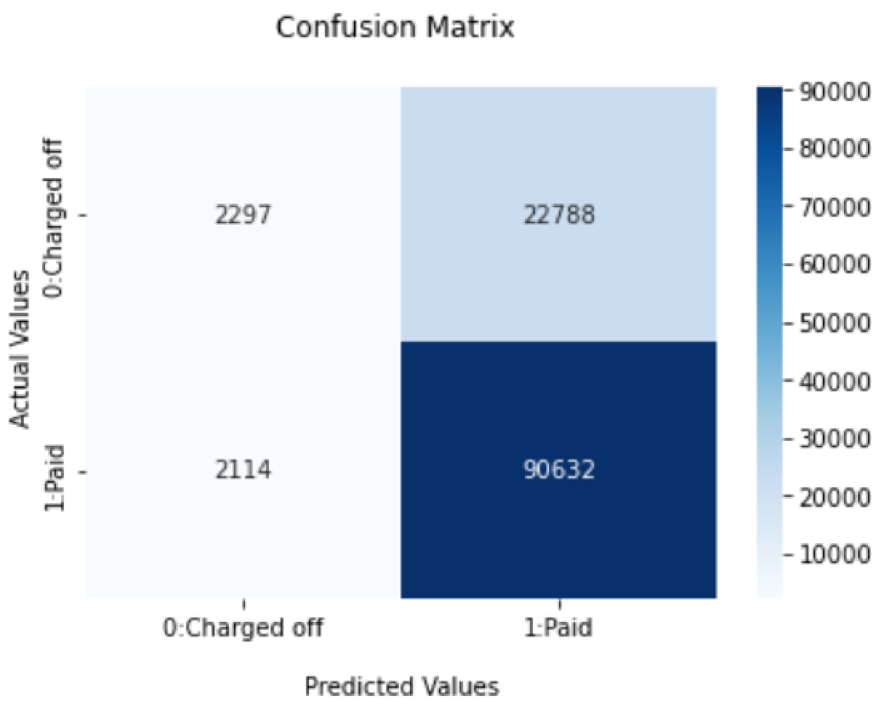


Classification report for test dataset

```
print(classification_report(y_test,test_pred))
```

```
              precision    recall  f1-score   support

           0       0.52      0.09      0.16     25085
           1       0.80      0.98      0.88     92746

    accuracy                           0.79    117831
   macro avg       0.66      0.53      0.52    117831
weighted avg       0.74      0.79      0.73    117831
```

Confusion matrix for test dataset



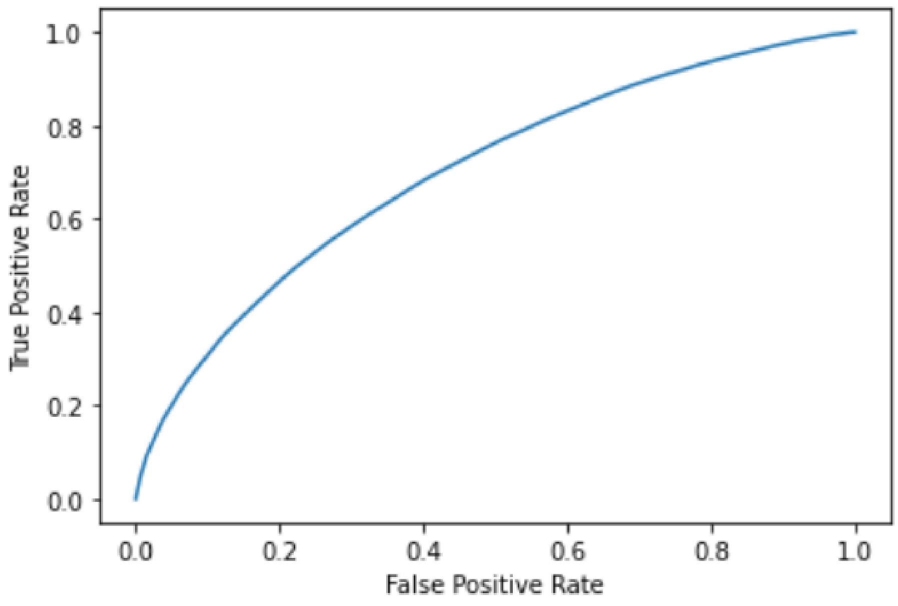
AUC Curve for test dataset

```
rf_prob_test = rf.predict_proba(X_test)[:,1]
```

```
roc_auc_score(y_test,rf_prob_test)
```

```
0.6984696538271504
```

ROC curve



XGBoost:

XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems. XGBoost builds upon supervised machine learning, decision trees, ensemble learning, and gradient boosting.

Decision trees create a model that predicts the label by evaluating a tree of if-then-else true/false feature questions and estimating the minimum number of questions needed to assess the probability of making a correct decision. Decision trees can be used for classification to predict a category, or regression to predict a continuous numeric value.

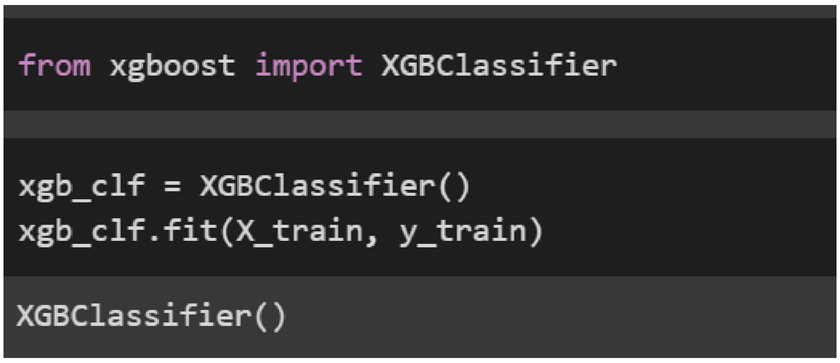
A Gradient Boosting Decision Trees (GBDT) is a decision tree ensemble learning algorithm similar to random forest, for classification and regression. Ensemble learning algorithms combine multiple machine learning algorithms to obtain a better model.

Both random forest and GBDT build a model consisting of multiple decision trees. The difference is in how the trees are built and combined.

The term “gradient boosting” comes from the idea of “boosting” or improving a single weak model by combining it with a number of other weak models in order to generate a collectively strong model. Gradient boosting is an extension of boosting where the process of additively generating weak models is formalized as a gradient descent algorithm over an objective function. Gradient boosting sets targeted outcomes for the next model in an effort to minimize errors. Targeted outcomes for each case are based on the gradient of the error (hence the name gradient boosting) with respect to the prediction.

XGBoost is a scalable and highly accurate implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms, being built largely for energizing machine learning model performance and computational speed. With XGBoost, trees are built in parallel, instead of sequentially like GBDT. It follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set.

Creating XGBoost for our dataset

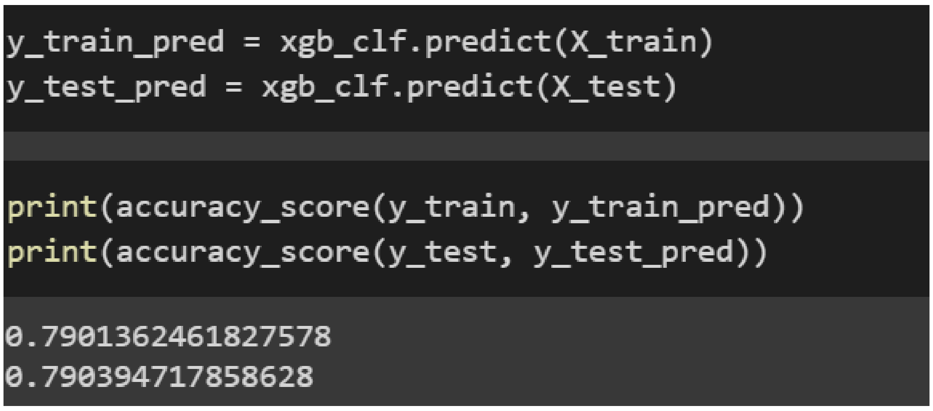


```
from xgboost import XGBClassifier
```

```
xgb_clf = XGBClassifier()
xgb_clf.fit(X_train, y_train)
```

```
XGBClassifier()
```

Predicting output for training and test dataset

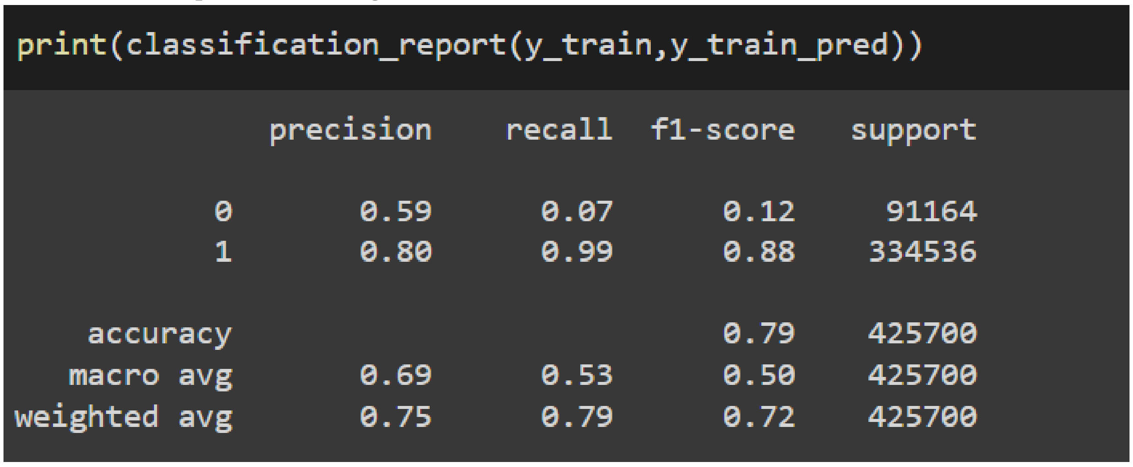


```
y_train_pred = xgb_clf.predict(X_train)
y_test_pred = xgb_clf.predict(X_test)
```

```
print(accuracy_score(y_train, y_train_pred))
print(accuracy_score(y_test, y_test_pred))
```

```
0.7901362461827578
0.790394717858628
```

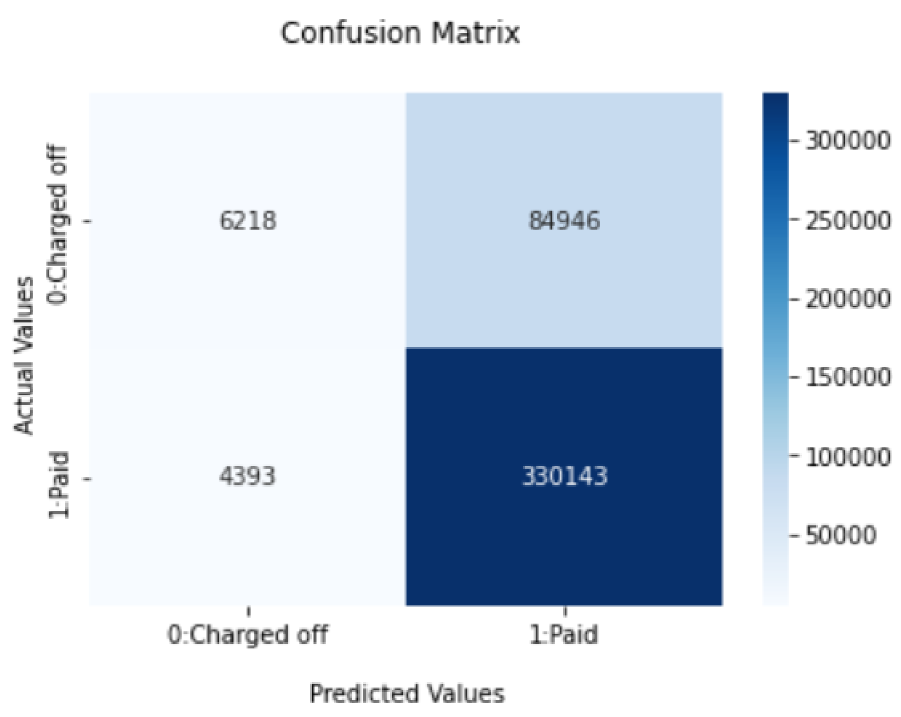
Classification report for training data



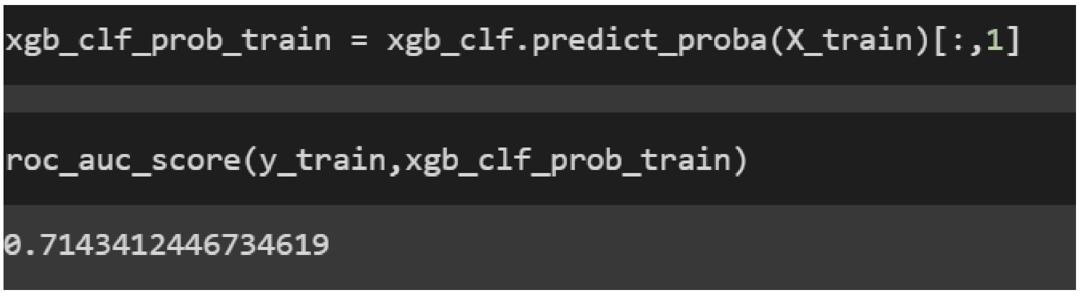
```
print(classification_report(y_train,y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.59      0.07      0.12     91164
           1       0.80      0.99      0.88    334536

    accuracy                           0.79    425700
   macro avg       0.69      0.53      0.50    425700
weighted avg       0.75      0.79      0.72    425700
```

Confusion matrix for training data

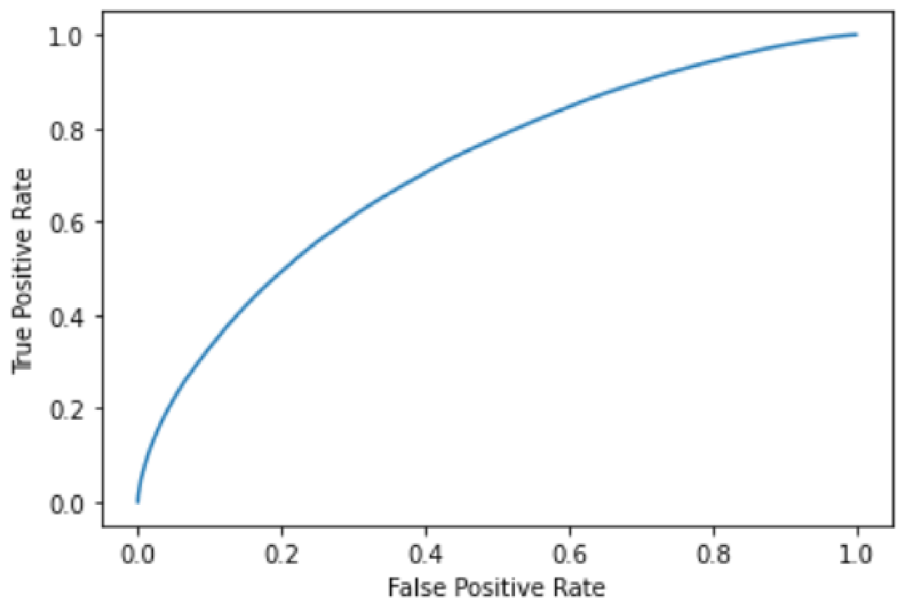


AUC score for training data

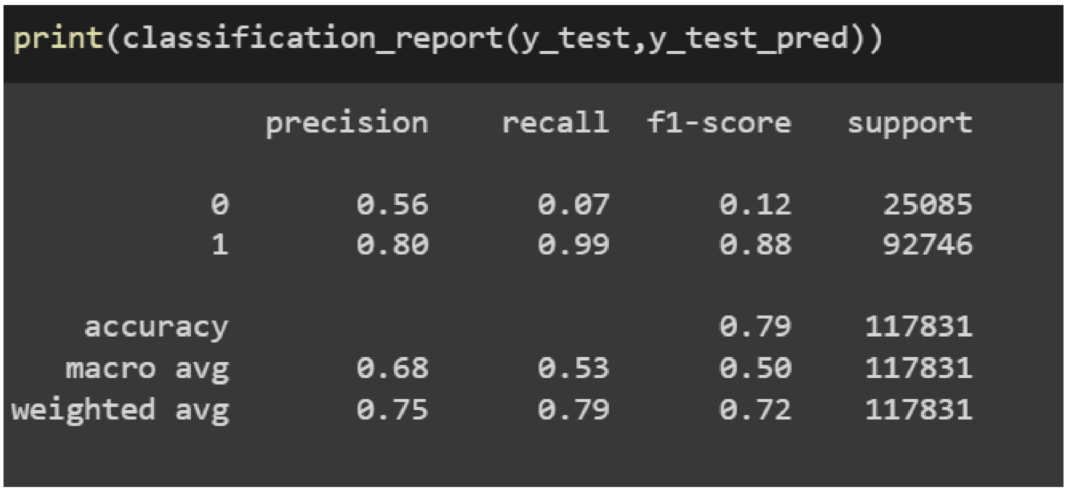


```
xgb_clf_prob_train = xgb_clf.predict_proba(X_train)[:,1]

roc_auc_score(y_train,xgb_clf_prob_train)

0.7143412446734619
```

ROC curve for training data

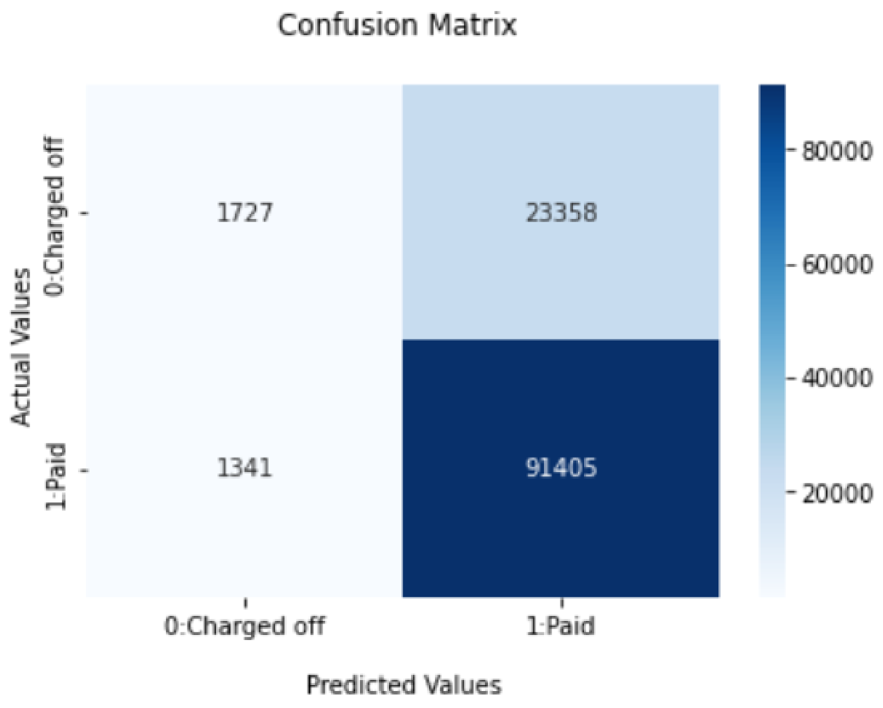


Classification report for test dataset

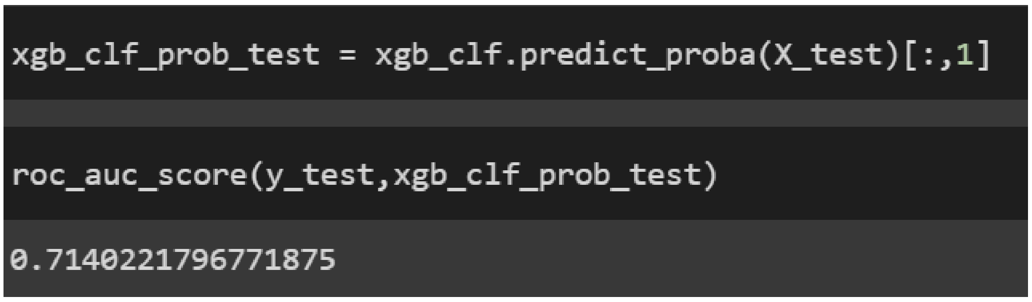


```
print(classification_report(y_test,y_test_pred))

              precision    recall  f1-score   support

           0       0.56      0.07      0.12     25085
           1       0.80      0.99      0.88     92746

    accuracy                           0.79    117831
   macro avg       0.68      0.53      0.50    117831
weighted avg       0.75      0.79      0.72    117831
```

Confusion matrix for test dataset

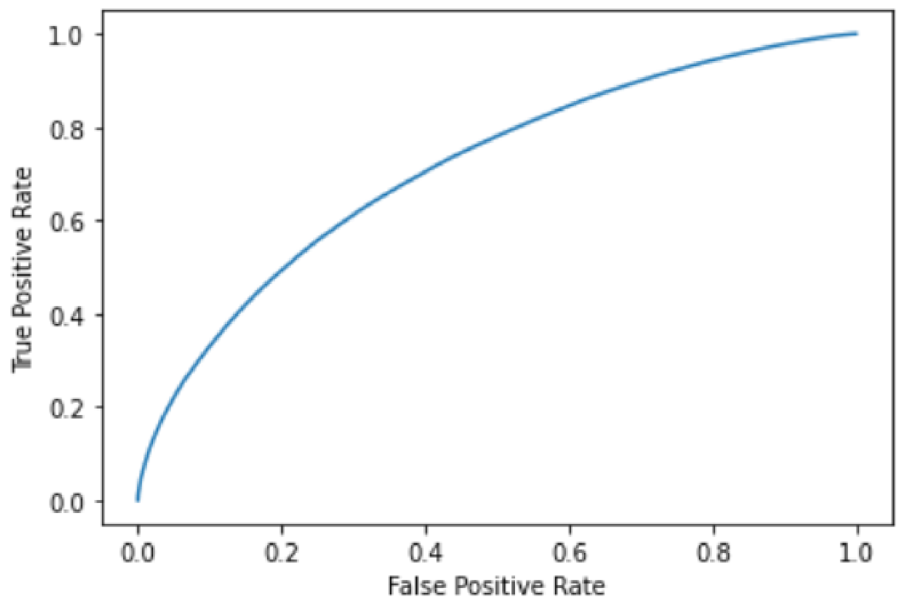


AUC score



```
xgb_clf_prob_test = xgb_clf.predict_proba(X_test)[:,1]

roc_auc_score(y_test,xgb_clf_prob_test)

0.7140221796771875
```

ROC Curve



While comparing all this model we can conclude that Random Forest is having higher accuracy of 78 percent for test data (which contains outliers). Also, we compared all other metrics and for our model deployment we are going to use Random Forest.

Chapter – VII

Model Deployment

Machine learning model deployment is the process of placing a finished machine learning model into a live environment where it can be used for its intended purpose. Models can be deployed in a wide range of environments, and they are often integrated with apps through an API so they can be accessed by end users.

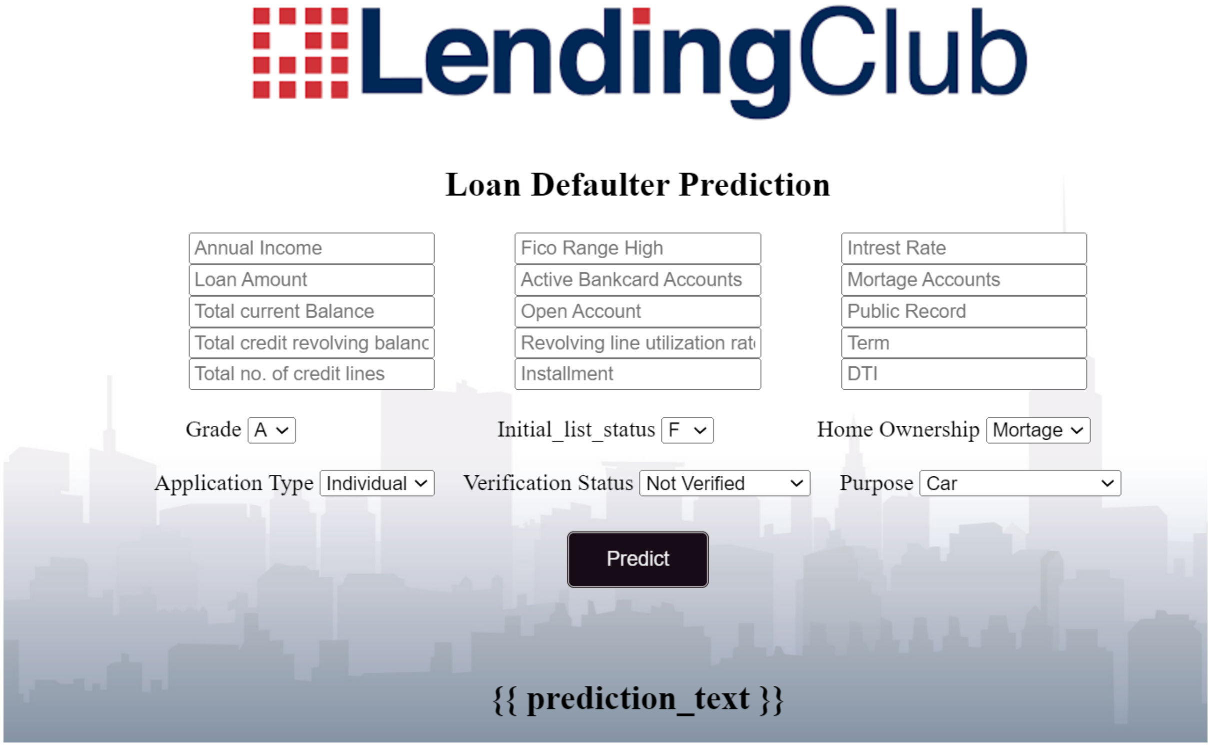
Flask is a Python-based micro framework used for developing small-scale websites. Flask is very easy to make Restful APIs using python.

Now we will design a web application where the user will input all the attribute values and the data will be given to the model, based on the training given to the model, the model will predict what should be the salary of the person whose details have been fed.

Deploy our model on flask:

HTML Form:

We first need to collect the data to predict the loan defaulter and then use the random forest model we build above to predict the borrower will pay their loan or not. Therefore, in order to collect the data, we create an HTML form which would contain all the different options to select from each attribute.


LendingClub

Loan Defaulter Prediction

Annual Income	Fico Range High	Intrest Rate
Loan Amount	Active Bankcard Accounts	Mortage Accounts
Total current Balance	Open Account	Public Record
Total credit revolving balanc	Revolving line utilization rat	Term
Total no. of credit lines	Installment	DTI

Grade A · Initial_list_status F · Home Ownership Mortage

Application Type Individual · Verification Status Not Verified · Purpose Car

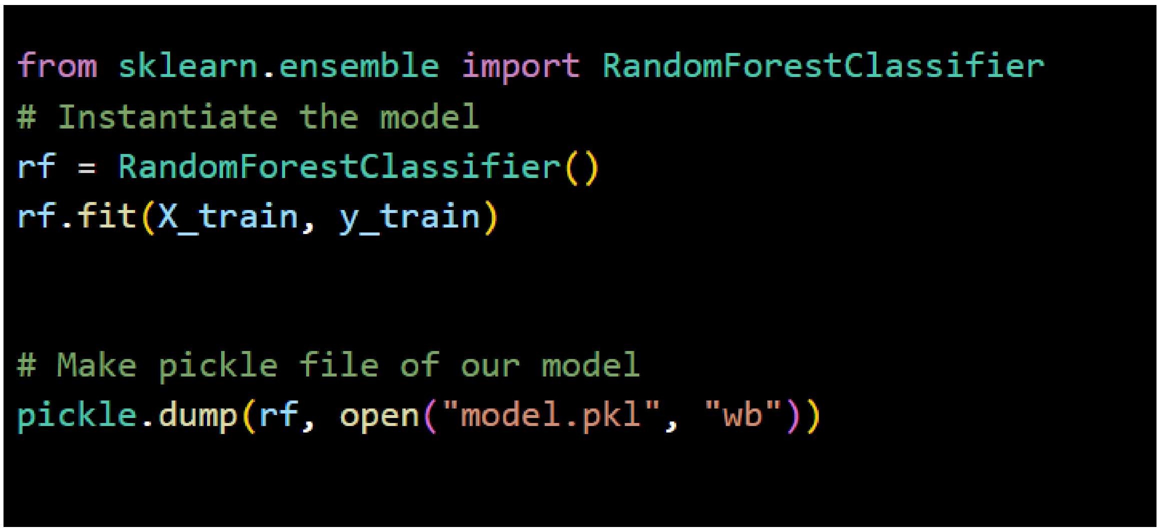
Predict

{{ prediction_text }}

Flask script:

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn’t enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy

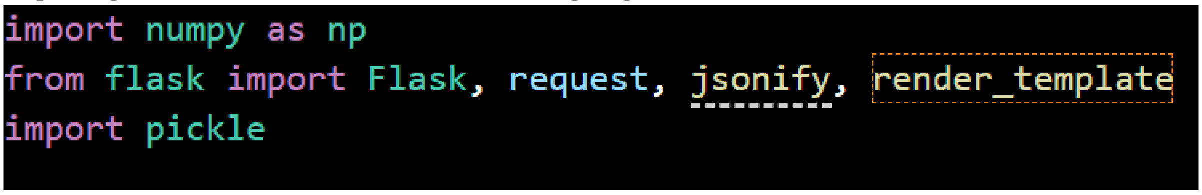


```python
from sklearn.ensemble import RandomForestClassifier
# Instantiate the model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)


# Make pickle file of our model
pickle.dump(rf, open("model.pkl", "wb"))
```

Here, our model.py is ready to train and save the model. Next we created app.py file.

In this file, we will use the flask web framework to handle the POST requests that we will get from the request.py.

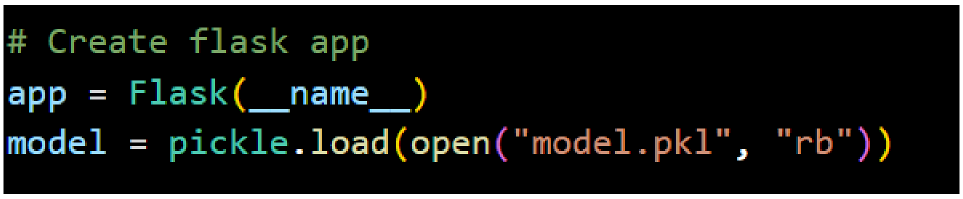
Importing the methods and libraries that we are going to use in the code.



```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
```

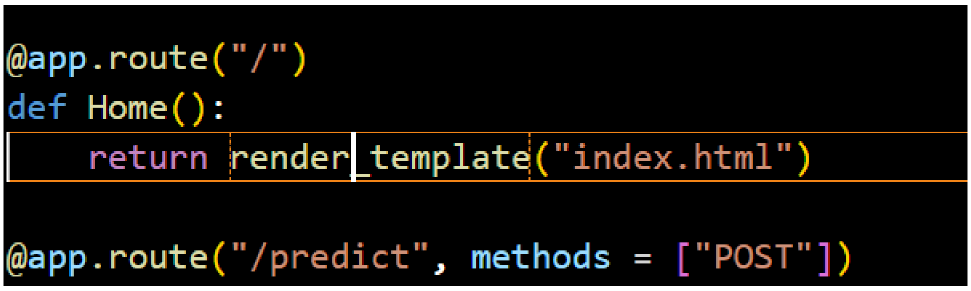
Here we have imported numpy to create the array of requested data, pickle to load our trained model to predict.

In the following section of the code, we have created the instance of the Flask() and loaded the model into the model.

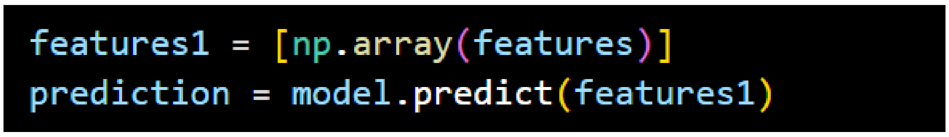


```python
# Create flask app
app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))
```

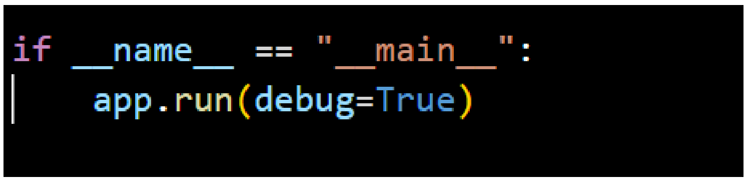
pickle.load() method loads the method and saves the deserialized bytes to model. Predictions can be done using model.predict().


```
@app.route("/")
def Home():
    return render_template("index.html")

@app.route("/predict", methods = ["POST"])
```

We create one define function to save all the input we are getting from that html file


```
 features1 = [np.array(features)]
 prediction = model.predict(features1)
```

Here, we have bounded /api with the method predict(). In which predict method gets the data from the html passed by the requestor. model.predict() method takes input from the html and converts it into 2D numpy array the results are stored into the variable named output and we return this variable after converting it into the html object using flasks .


```
if __name__ == "__main__":
    app.run(debug=True)
```

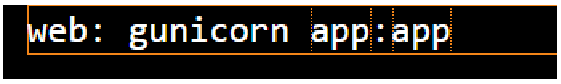
Finally, we will run our server by following code section. Here I have set debug=True since if we get any error, we can debug it and solve it.

We are creating another file, requirements.txt where we were stored all the libraries we used in this deployment and its version.

Next, we are created procfile in that we used Gunicorn

Gunicorn is one of many WSGI server implementations, but it's particularly important because it is a stable, commonly-used part of web app deployments that's powered some of the largest Python-powered web applications in the world, such as Instagram.

Gunicorn implements the PEP3333 WSGI server standard specification so that it can run Python web applications that implement the application interface. For example, if you write a web application with a web framework such as Django, Flask or Bottle, then your application implements the WSGI specification.

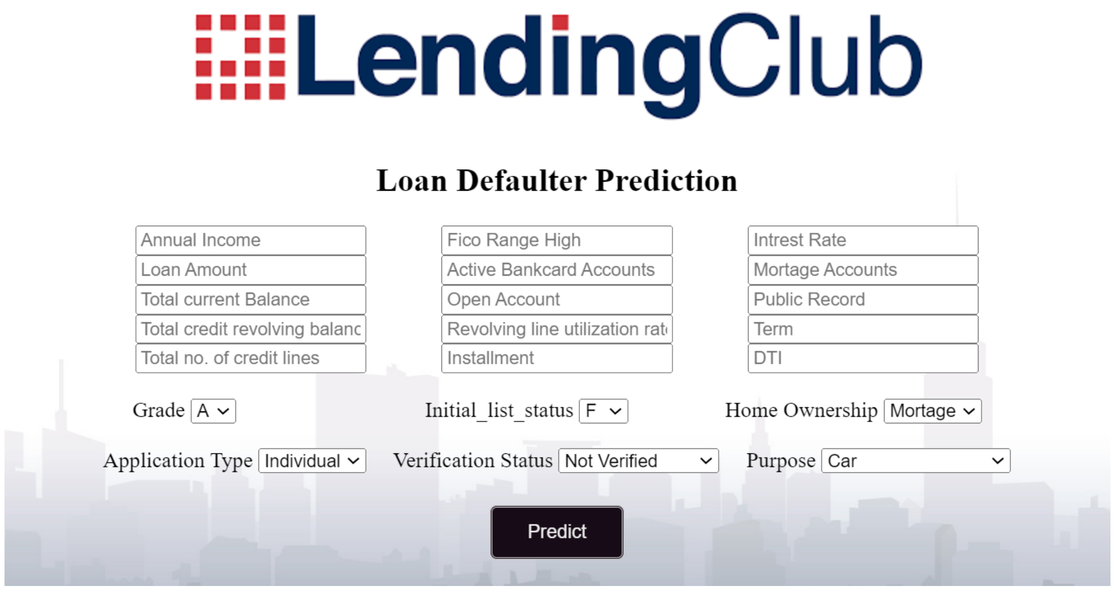

```
web: gunicorn app:app
```

We uploaded all the model files on GitHub. Then we will connect that file in Heroku to create our webserver.

Once files had uploaded to the GitHub repository, we are now ready to start deployment on Heroku.

- After signup on heroku.com then click on Create new app.
- Enter App name and region.
- Connect to your GitHub repository where code is uploaded.
- Deploy branch and create our webserver

The app is published at URL: https://lending-club-default-pred.herokuapp.com/



Loan Defaulter Prediction

Grade A

Initial_list_status F

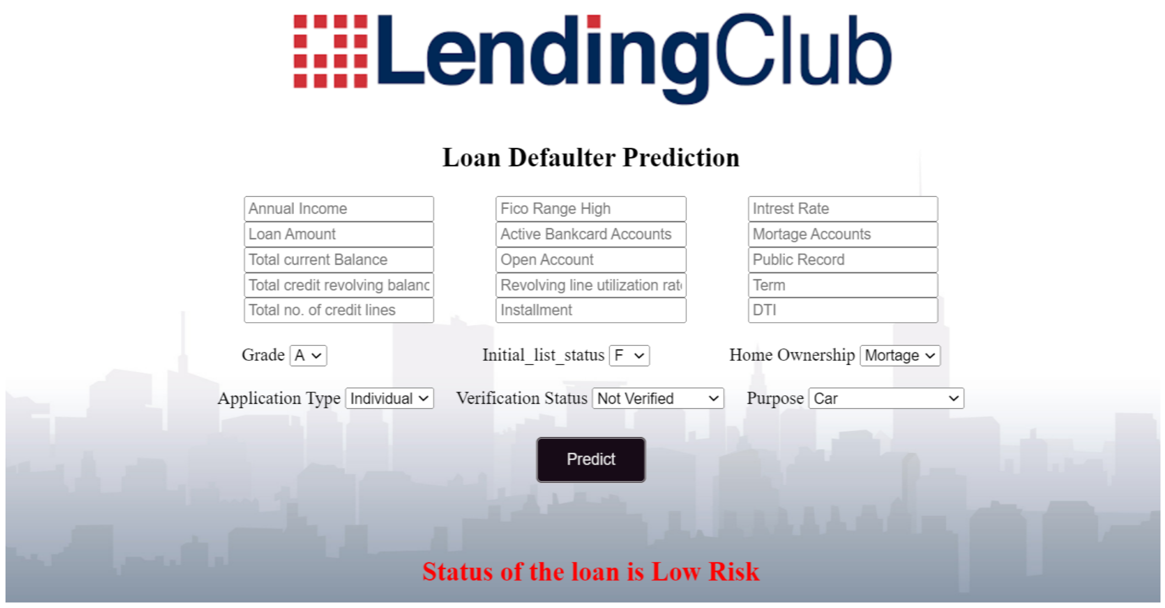
Home Ownership Mortage

Application Type Individual

Verification Status Not Verified

Purpose Car

Let’s enter some values to check our model is working properly.



Loan Defaulter Prediction

Grade A

Initial_list_status F

Home Ownership Mortage

Application Type Individual

Verification Status Not Verified

Purpose Car

Status of the loan is Low Risk

Chapter – VIII

Conclusion

From this research, financial technology like online peer-to-peer (P2P) lending is the current trend that already disturbed the finance industry. However, they are facing the great risk if more borrower loan becomes default, then it’ll be more difficult for them. Because they didn’t give there, it’s the money of their investors. So, they need to analyse each application before giving them the loan, but it'll be more difficult to do that.

When we are doing EDA, we found out some risk categories are there. For those risky borrowers we need to analyse their profile and then choose the appropriate decisions based on their credit history. Here we listed a few risky borrowers, who are all come under certain categories.

1. Loans having higher interest rate have more defaulters. Check the background of applicant thoroughly if interest rate is high.

2. When the purpose is debt consolidation check applicant thoroughly as it has high tendency to default.

Tree based models have relatively better accuracy as compared to other models and that’s why we see them as better models. When Lending Club uses the model, we want to assign each loan a probability for Lending Club to make their own decision, not to just automatically accept or deny loans.

References:

- https://www.lendingclub.com/
- https://www.analyticsvidhya.com/blog/2021/08/data-preprocessing-in-data-mining-a-hands-on-guide/
- https://www.digitalocean.com/community/tutorials/exploratory-data-analysis-python
- https://www.datacamp.com/tutorial/random-forests-classifier-python
- https://www.datatechnotes.com/2019/07/classification-example-with.html
- https://www.datacamp.com/tutorial/understanding-logistic-regression-python
- https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761