Technocolabs Data Science Internship

DA Project Report

TITLE: Lending Club's Loan Repayment Prediction

AIM:

The main purpose of this project is to develop machine learning models (Regularized Logistic Regression and Random Forest) on Lending Club's data and deploy them for wider general-purpose usage.

ABSTRACT:

Traditionally, loan-level risk is measured as credit risk—the probability of default to measure the expected loss. Using machine learning techniques, we modelled credit risk and expected payoff maximization on the ROC, to help Lending Club optimize their risk. The models used here are Regularized Logistic Regression and Random Forest .This project aims to analyze the loans that were paid off in full or charged off.

INTRODUCTION:

Lending Club's dataset contains extensive information of their customer's loan status and many other metrics which help in the analysis of its data. The goal here is to build a model using this data and use that model to accurately predict whether the loans were either fully paid or charged off.

OVERVIEW:

1. Examine and Clean the Dataset
2. Visualize the dataset for gaining clarity
3. Build and Train the model
4. Visualize the Model's end Report

DATASET:

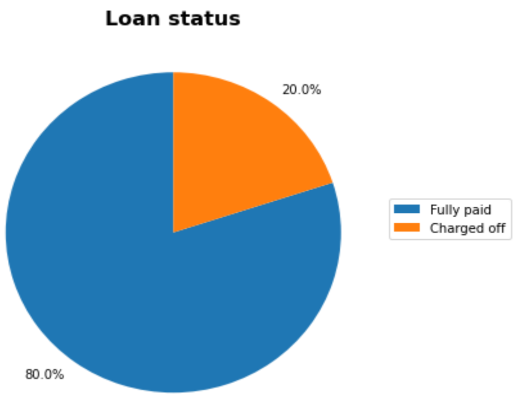
The Lending Club Dataset Contains 2260701 rows by default, which detail the number of customers and it has 151 columns which contain all details with respect to their lending club loan data requirements.

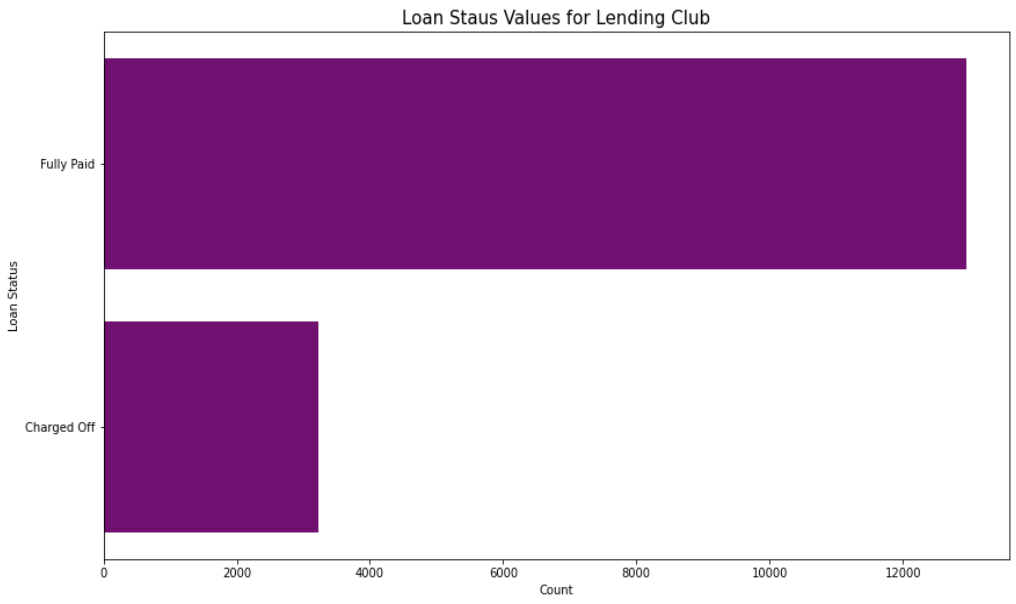
1) EXAMINE AND CLEAN THE DATASET:

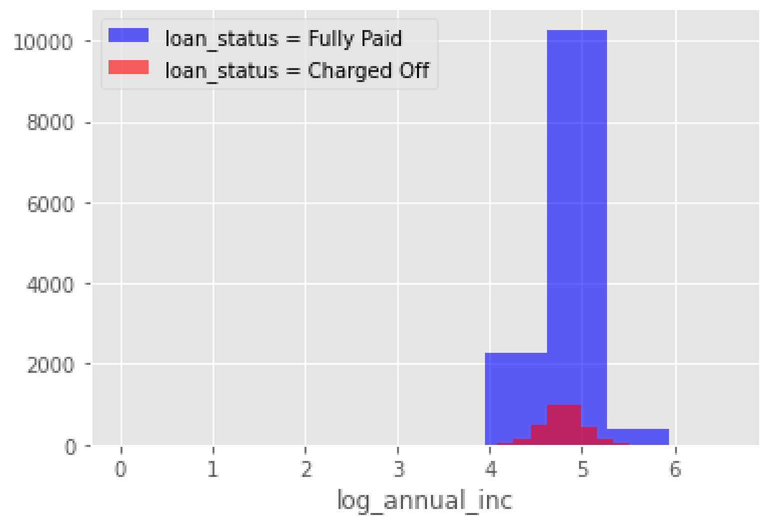
- The dataset is readily available from the bank's own website and via Kaggle.com as well. The data is in the form of .csv(comma-separated values) file. These files need to be loaded in the folder which contains the ipython notebook which will be used for analysis and model building.
- The dataset is then analysed for any discrepancies(mostly null values) and the percentage of null values in every column must be observed.
- Then to analyze the loan status of the dataset, we label the values "Charged Off" as 0 and "Fully Paid" as 1 for proper classification

2) VISUALIZING THE DATASET:

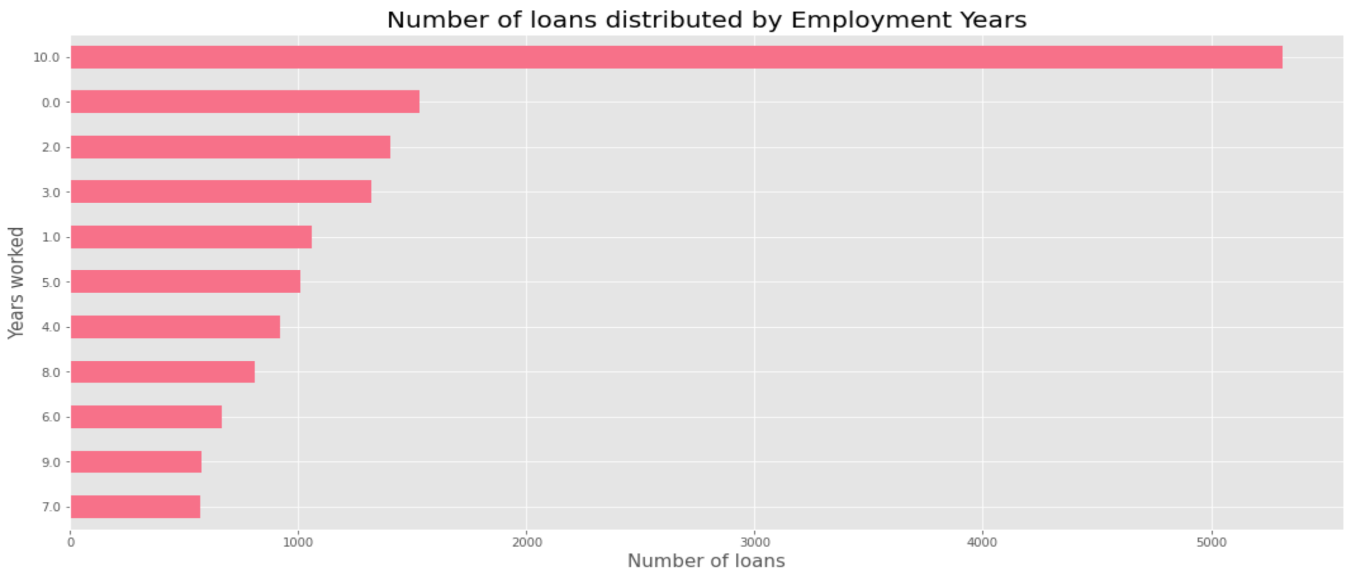
- Here we visualize the dataset parameters using matplotlib and seaborn modules
- The charts produced are:



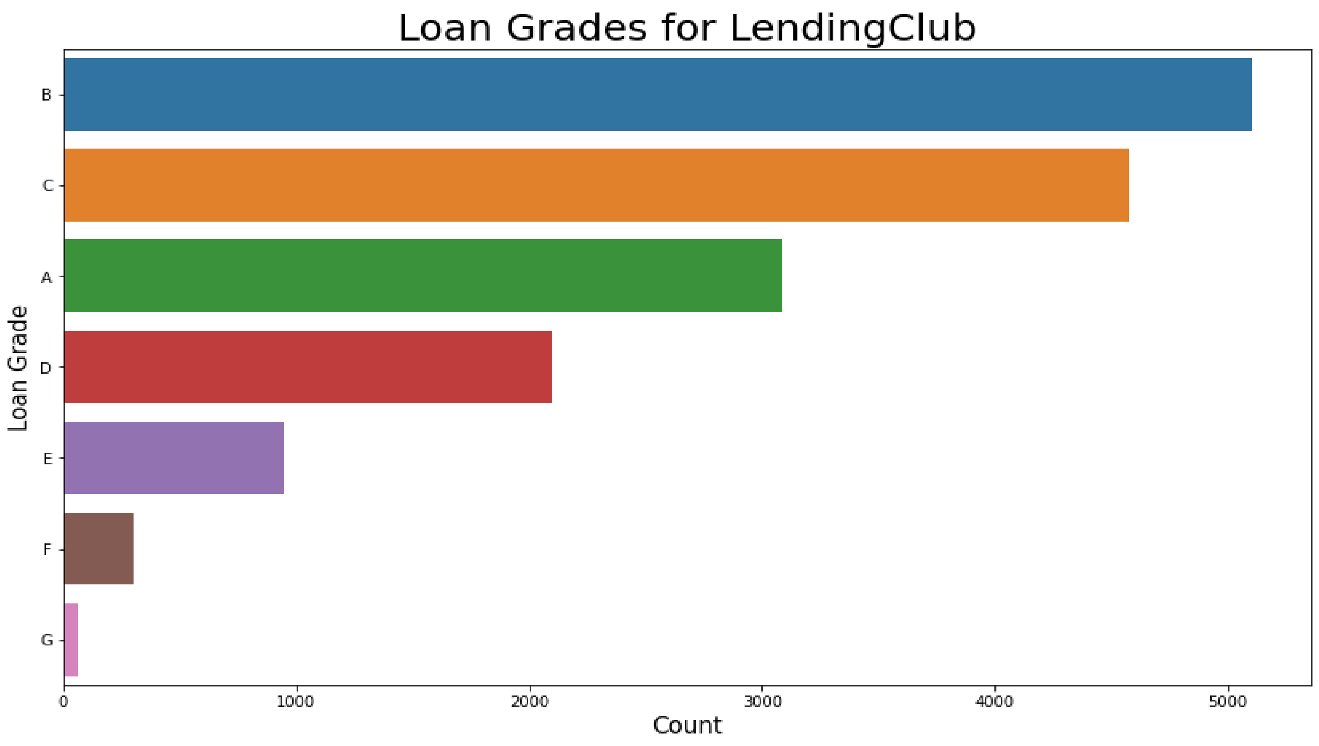




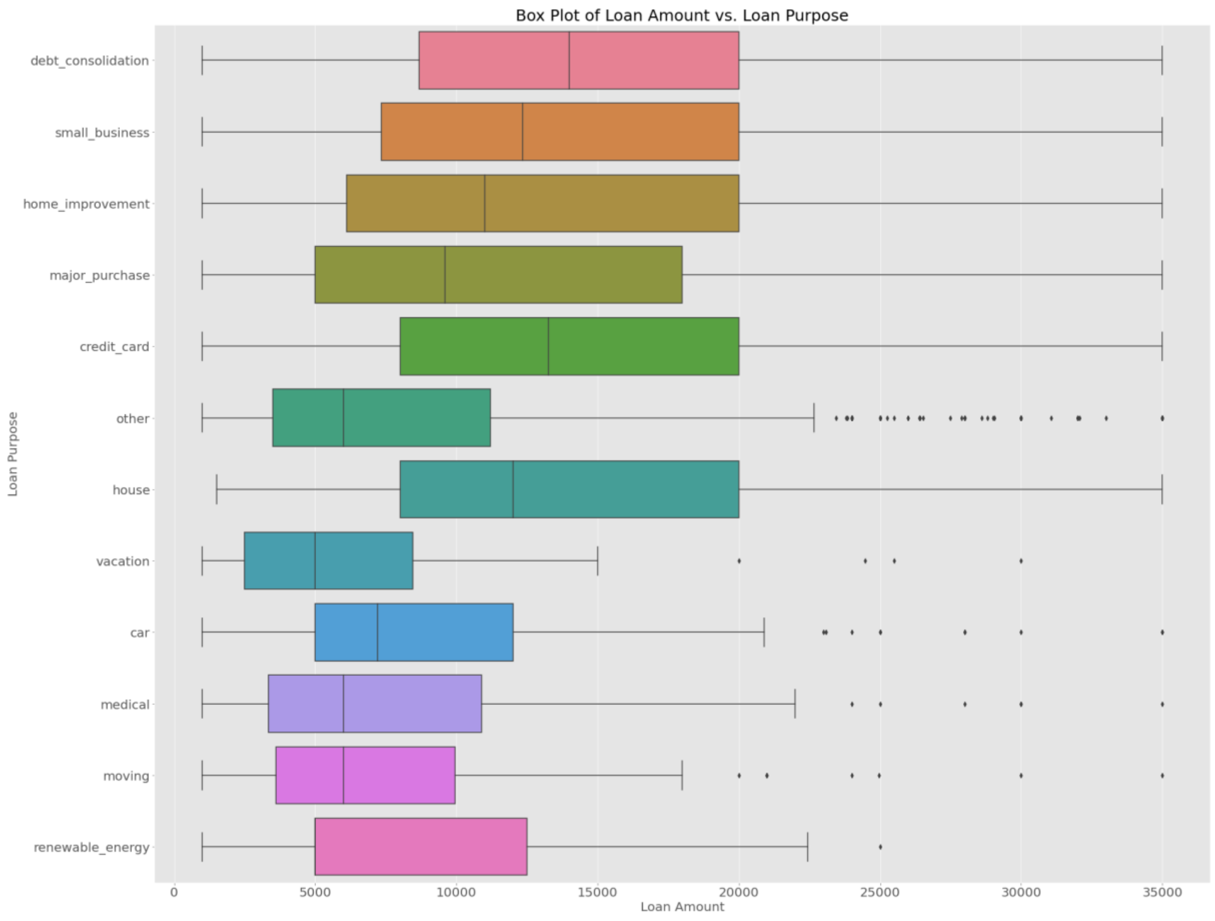
loan_status = Fully Paid
loan_status = Charged Off
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
6
log_annual_inc



Number of loans distributed by Employment Years
Years worked
10.0
0.0
2.0
3.0
1.0
5.0
4.0
8.0
6.0
9.0
7.0
0
1000
2000
3000
4000
5000
Number of loans



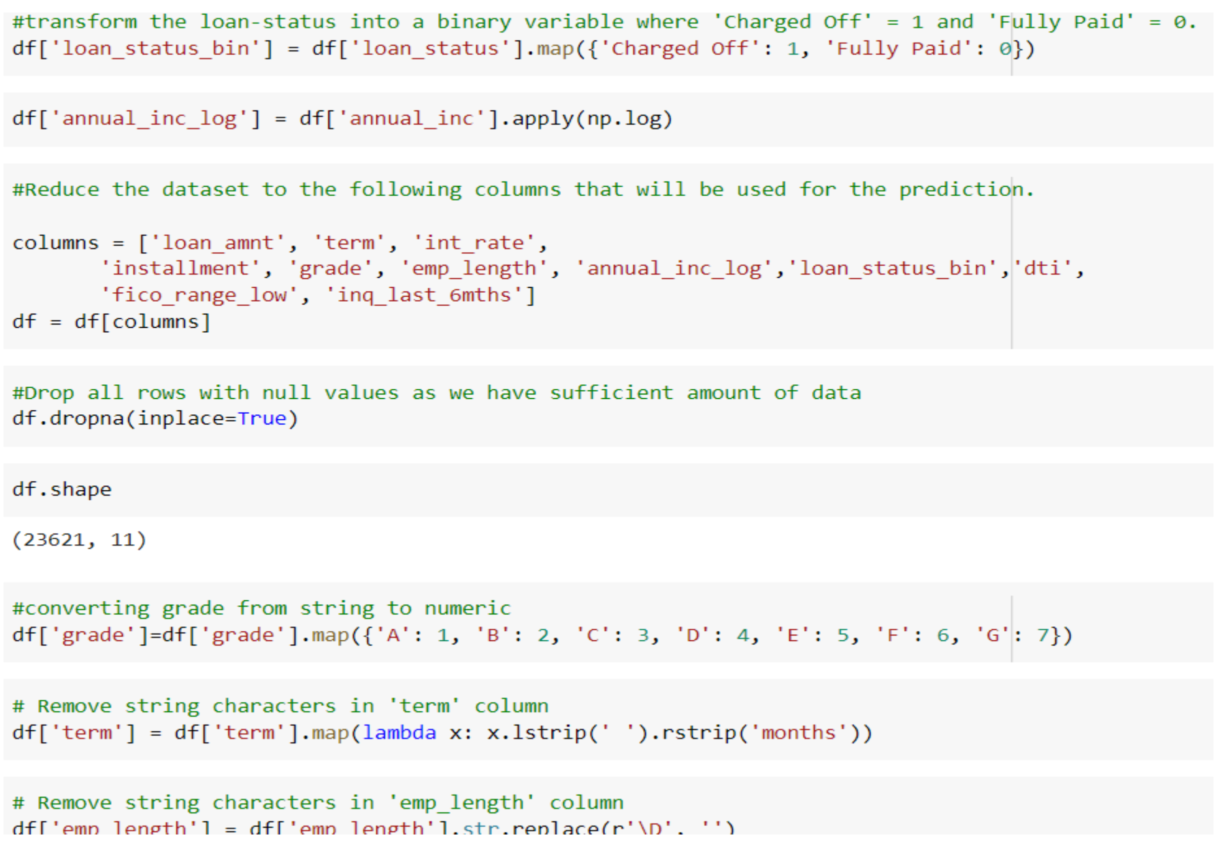
Loan Grades for LendingClub
Loan Grade
B
C
A
D
E
F
G
0
1000
2000
3000
4000
5000
Count



3) BUILDING AND TRAINING THE MODEL:

- The regularized logistic regression model needs to built using numeric data only. Hence the first step is to encode all the columns with non-numeric data types(they are mostly object data type) and assign numeric labels.
- Regularized Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of categorical dependent variable.In logistic regression, the dependent variable is a binary or Multinomial, which involves having more than one category.
  - Regularized Logistic Regression Assumptions:
  - The independent variables should be independent of each other.
    That is, the model should have little or no multicollinearity.
  - The independent variables are linearly related to the log odds.
  - Logistic regression requires quite large sample
- In the next step we split the data into training and testing sets using the train_test_split function.

- After obtaining the train and test sets we fit the data into the Regularized Logistic Regression model and let it train. After the training ends, we use the model to test it on the test set and obtain predicted values.
- **Create Test and Train Dataset**



```
#transform the loan-status into a binary variable where 'Charged Off' = 1 and 'Fully Paid' = 0.
df['loan_status_bin'] = df['loan_status'].map({'Charged Off': 1, 'Fully Paid': 0})
```

```
df['annual_inc_log'] = df['annual_inc'].apply(np.log)
```

```
#Reduce the dataset to the following columns that will be used for the prediction.

columns = ['loan_amnt', 'term', 'int_rate',
       'installment', 'grade', 'emp_length', 'annual_inc_log','loan_status_bin','dti',
       'fico_range_low', 'inq_last_6mths']
df = df[columns]
```

```
#Drop all rows with null values as we have sufficient amount of data
df.dropna(inplace=True)
```

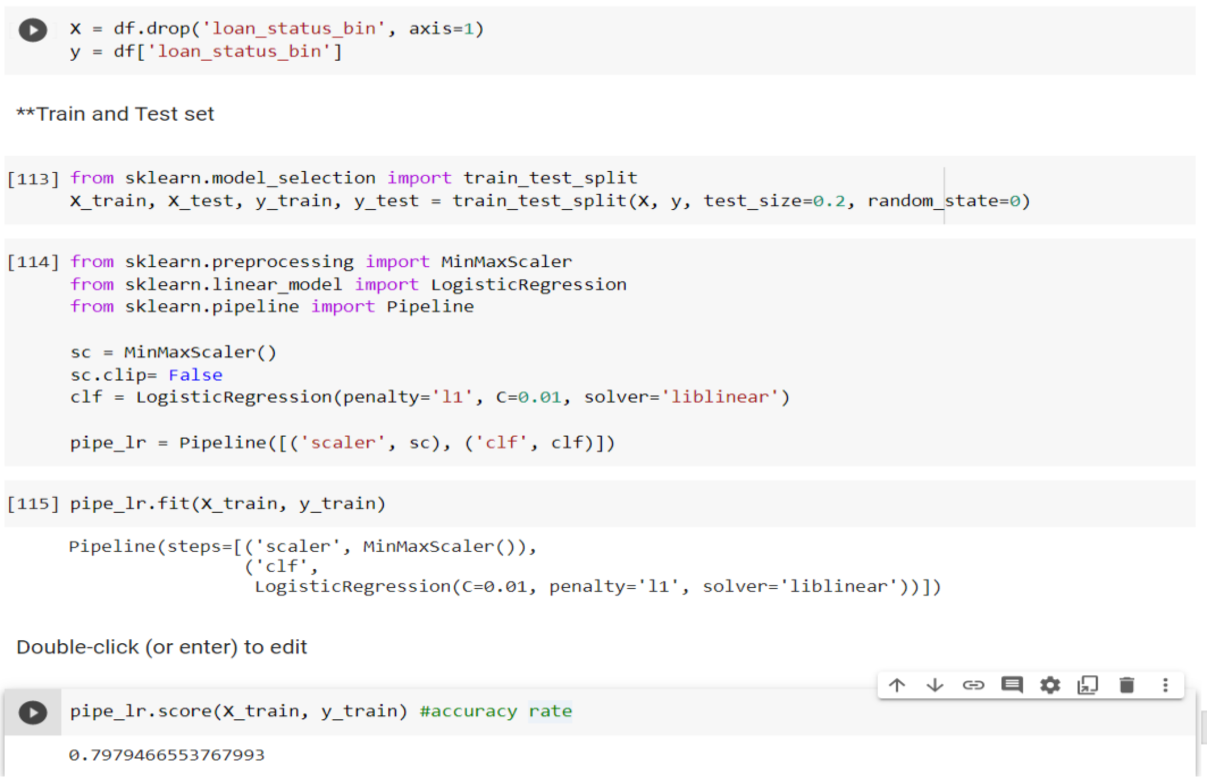
```
df.shape
```

```
(23621, 11)
```

```
#converting grade from string to numeric
df['grade']=df['grade'].map({'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7})
```

```
# Remove string characters in 'term' column
df['term'] = df['term'].map(lambda x: x.lstrip(' ').rstrip('months'))
```

```
# Remove string characters in 'emp_length' column
df['emp_length'] = df['emp_length'].str.replace(r'\D', '')
```



```
X = df.drop('loan_status_bin', axis=1)
y = df['loan_status_bin']
```

**Train and Test set

```
[113] from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

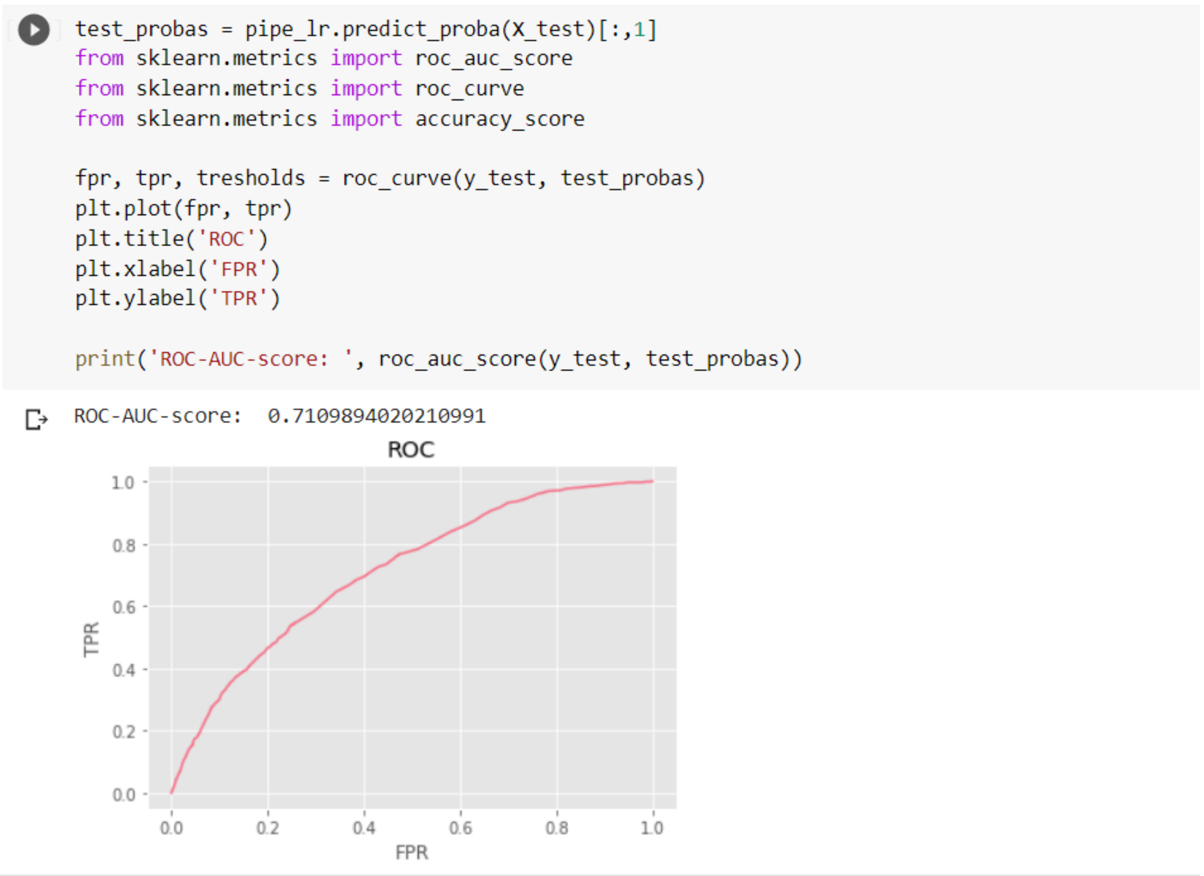
```
[114] from sklearn.preprocessing import MinMaxScaler
      from sklearn.linear_model import LogisticRegression
      from sklearn.pipeline import Pipeline

      sc = MinMaxScaler()
      sc.clip= False
      clf = LogisticRegression(penalty='l1', C=0.01, solver='liblinear')

      pipe_lr = Pipeline([('scaler', sc), ('clf', clf)])
```

```
[115] pipe_lr.fit(X_train, y_train)

      Pipeline(steps=[('scaler', MinMaxScaler()),
                      ('clf',
                       LogisticRegression(C=0.01, penalty='l1', solver='liblinear'))])
```

Double-click (or enter) to edit

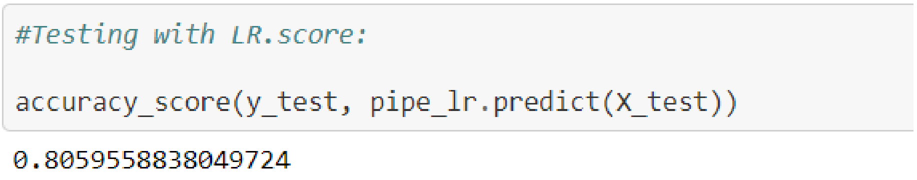
```
pipe_lr.score(X_train, y_train) #accuracy rate

0.7979466553767993
```

4) Visualize the ROC curve of this model:



```
test_probas = pipe_lr.predict_proba(X_test)[:,1]
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import accuracy_score

fpr, tpr, tresholds = roc_curve(y_test, test_probas)
plt.plot(fpr, tpr)
plt.title('ROC')
plt.xlabel('FPR')
plt.ylabel('TPR')

print('ROC-AUC-score: ', roc_auc_score(y_test, test_probas))
```

```
ROC-AUC-score:  0.7109894020210991
```



```
#Testing with LR.score:

accuracy_score(y_test, pipe_lr.predict(X_test))
```

```
0.8059558838049724
```

- Here as we can see, the model is approximately 81% accurate.

DEPLOYMENT OF THE TRAINED AND TESTED MODEL:

- Deploying a machine learning model, known as model deployment, simply means to integrate a machine learning model and integrate it into an existing production environment, where it can take in an input and return an output.
- Deployment can be carried out by both flask and streamlit applications. Here we used streamlit.
- In Streamlit, first we saved our model using in a pickle file format(.pkl) and loaded it into a new python file. There we specified html parameters for the look of the end application and created created a function which used this model to display the predictions. The predictions would be displayed as either “Charged Off” or “Fully Paid”.
- Then we finally deployed our streamlit model using Heroku for open world use.
- Deployed Online url: https://predicting-loan.herokuapp.com