

Loan Default Prediction

Banking Analytics

Umesh Arora

Loan Defaulter Prediction - Banking Analytics

Problem Statement

To develop a predictive model that will facilitate the approval of loans, with the intention of minimizing issuance of bad loans and maximizing interest income for the lenders.

Loan Defaulter Prediction

Approach

- Defining the problem statement and objectives
- Cleaning and engineering of data
- Exploratory Data Analysis (EDA)
- Further Cleaning and engineering of data
- Correlation Matrix
- Model Building
- Results
- Conclusion

Defining the objective

I obtained the Lending Club dataset from Kaggle which came with a data dictionary. This was rather fortunate, as it meant that I would not need to make sense of the data entirely from scratch. There were a number of features which could not be used (I will elaborate further on this in the following sections) but I was able to verify that the dataset had the relevant features required for predicting whether or not the loan would be repaid.

My goals for the project is as below :

- *Classification model to identify non-performing loans*

Cleaning and Engineering of Data

There were 96783 records and 150 features to start with.

The target feature “**loan_status**” had following unique values :

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

```
↳ array(['Charged Off', 'Fully Paid', 'Current', 'Late (16-30 days)',  
        'Late (31-120 days)', 'In Grace Period', 'Default', nan],  
        dtype=object)
```

```
[ ] data['loan_status'].value_counts()
```

```
↳ Fully Paid          71659  
   Charged Off        15248  
   Current            9233  
   Late (31-120 days)   352  
   In Grace Period     142  
   Default             98  
   Late (16-30 days)   47  
   ..                 ..
```

Cleaning and Engineering of Data

From the investor's perspective, I am interested in trying to predict which loans will be paid off on time and which ones won't be. Only the **Fully Paid** and **Charged Off** values describe the final outcome of the loan. The other values describe loans that are still ongoing and where the jury is still out on if the borrower will pay back the loan on time or not. While the Default status resembles the Charged Off status, in Lending Club's eyes, loans that are charged off have essentially no chance of being repaid while default ones have a small chance.

Cleaning and Engineering of Data

I kept into consideration only the records with **loan_status** as **Fully Paid** and **Charged Off** and got rid of all the remaining records as a result, the no of records were reduced from 96783 to 86907

```
[ ] data = data[( data['loan_status'] == "Fully Paid") | (data['loan_status'] == "Charged Off")]
```

```
[ ] data['loan_status'].value_counts()
```

```
1    71659
0    15248
Name: loan_status, dtype: int64
```

Cleaning and Engineering of Data

Analyzing the data , I found that some of the features had null values predominantly. I decided to retain the features that had less than 30% null values which resulted in the reduction of features from 150 to 81

```
[ ] data = data.loc[:, data.isin([' ', 'NULL', 0 , np.nan]).mean() < .3]
```

```
[ ] print("Database has {} obserwations (customers) and {} columns (attributes)".format(data.shape[0],data.shape[1]))
```

```
↳ Database has 86907 obserwations (customers) and 81 columns (attributes).
```

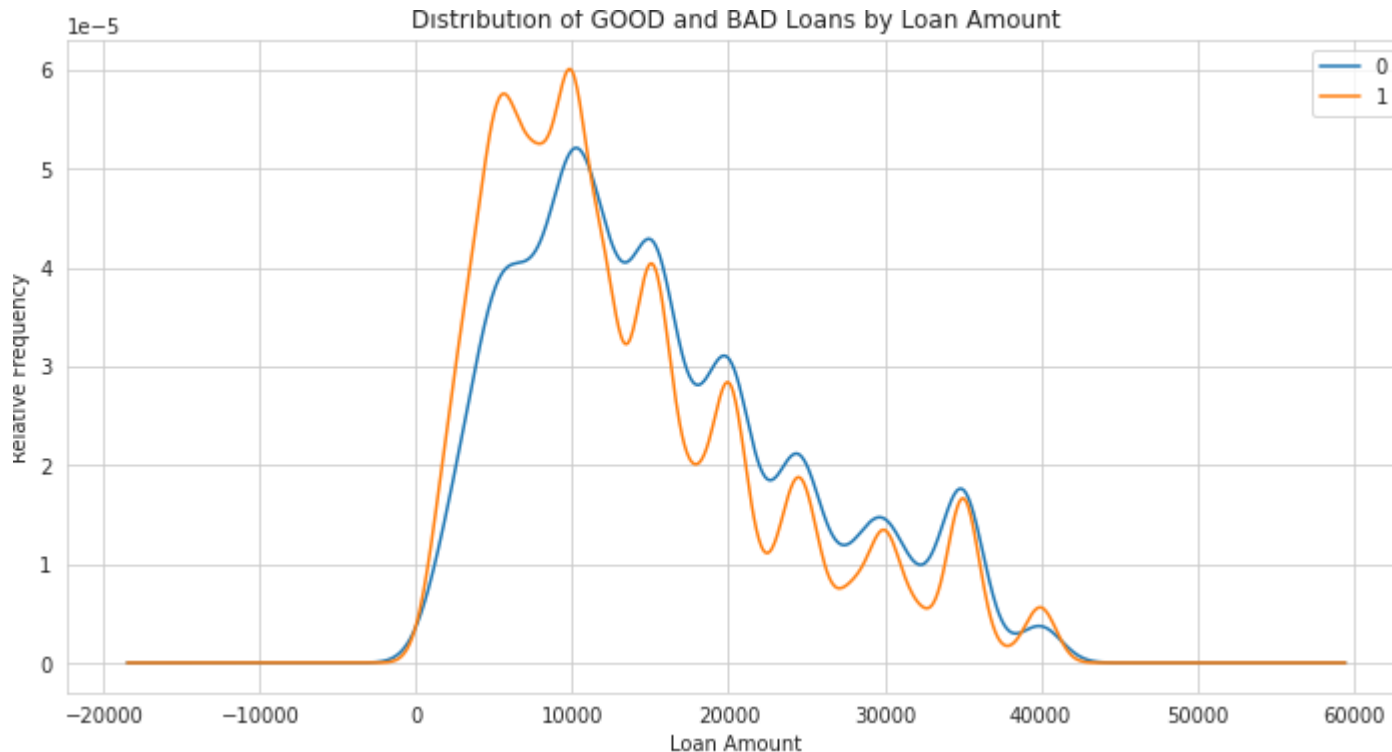

Exploratory Data Analysis

The EDA process was meant to uncover patterns, trends and initial insights that could highlight relevant features for further investigation in the modelling process. My initial hypothesis was that the features such as Annual Income, Interest Rate, Credit Grade and Loan Amount would have a bearing on whether or not the borrower defaulted.

I found that the Loan Amount did not really have a bearing on whether the borrower would eventually default. My hunch was that this is because of the nature of loans given out by this institution, which restricts the amount of loan and the actual values that can be given out.

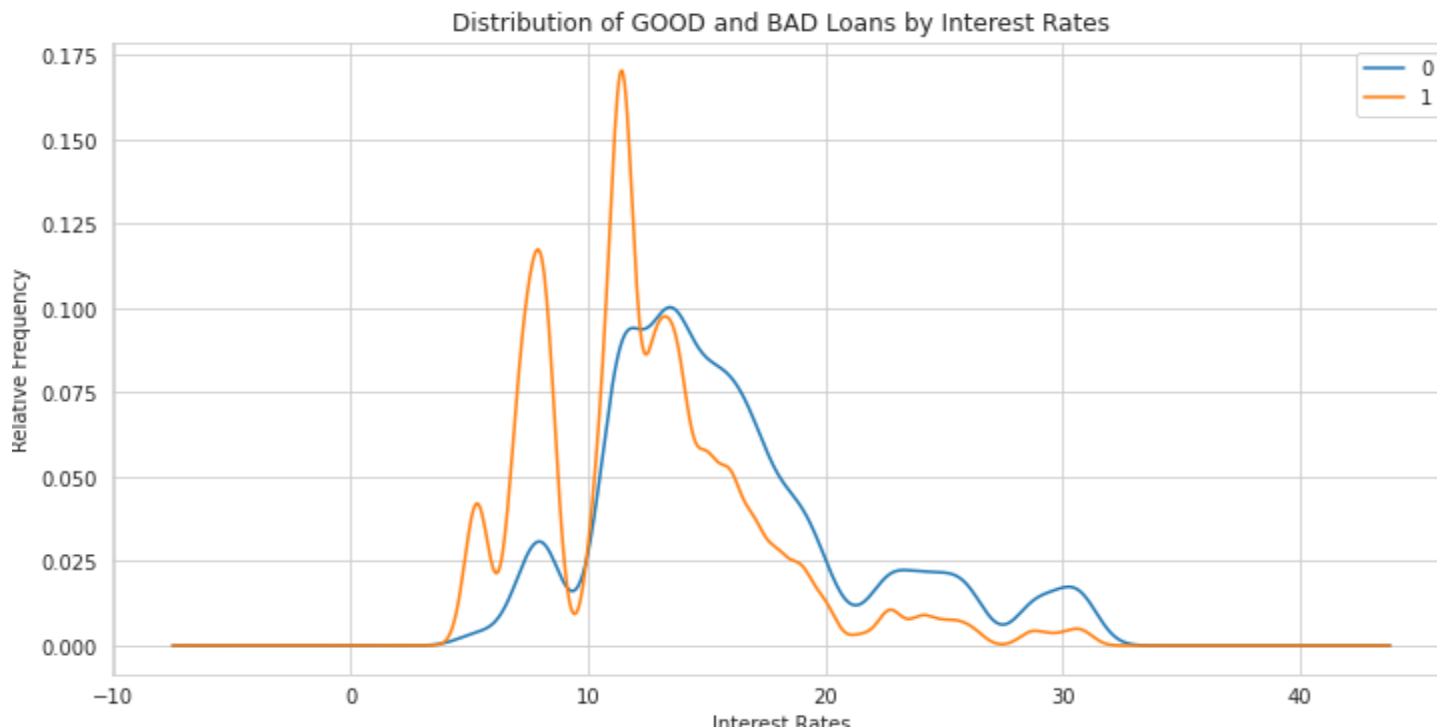
Exploratory Data Analysis

Orange Line signifies Fully Paid , Blue Line signifies Defaulted



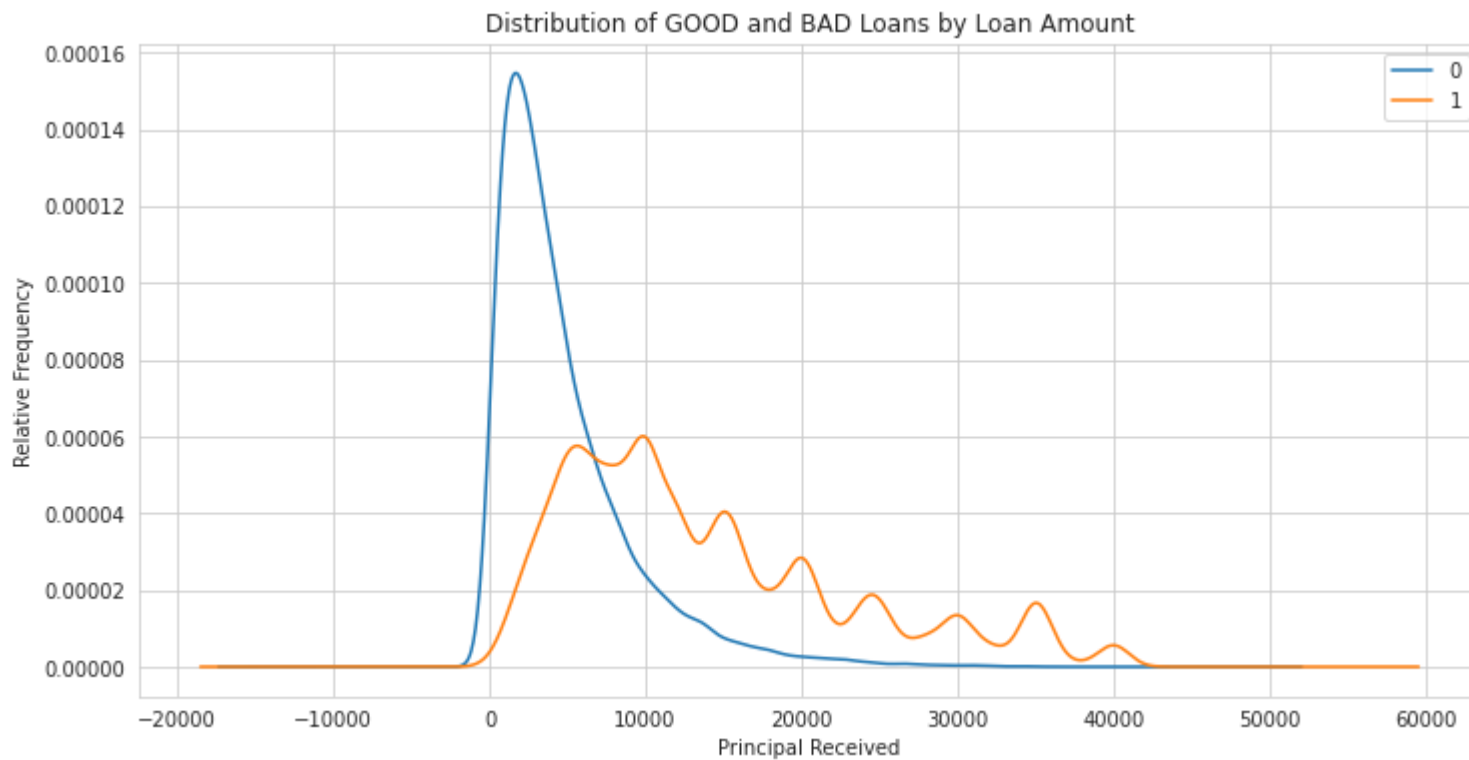
Exploratory Data Analysis

Interest Rates for the loan did not appear to have a bearing on whether or not the borrower would default. Orange Line signifies Fully Paid , Blue Line signifies Defaulted.



Exploratory Data Analysis

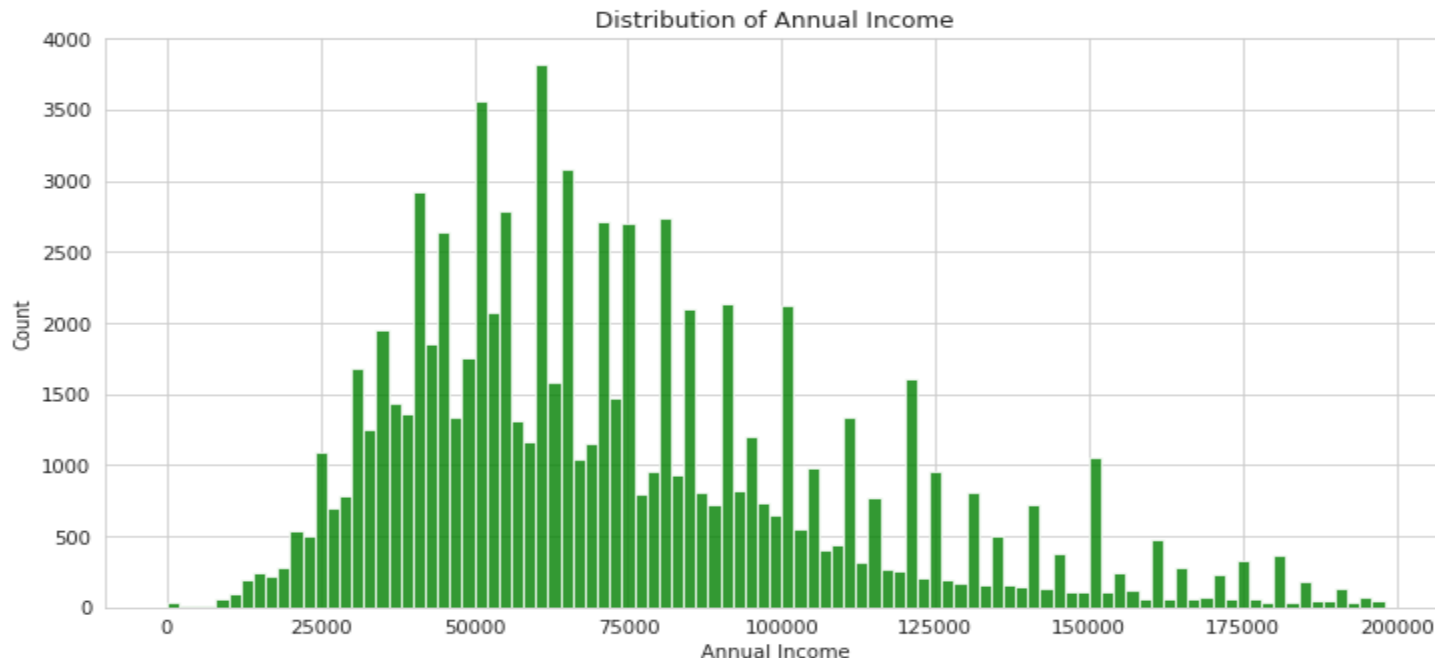
Principal Amount Received appeared to have a bearing on whether or not the borrower would default. Orange line signifies Fully Paid , Blue Line signifies Defaulted



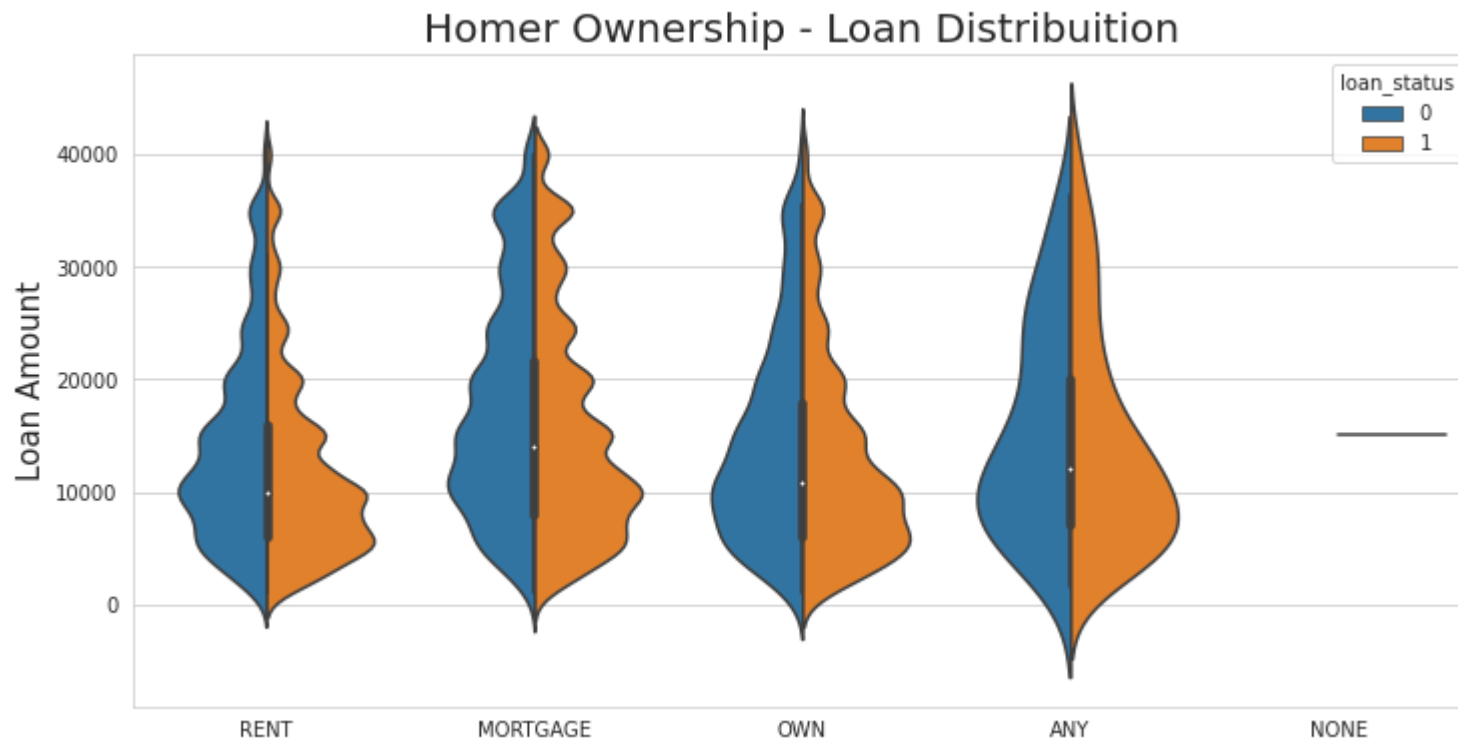
Exploratory Data Analysis

For the sake of brevity, I am listing some of the other visualisations obtained from the EDA process below.

Unless otherwise specified, 1's correspond to loans associated with Good Loans, whilst 0's correspond to Defaulted Loans.

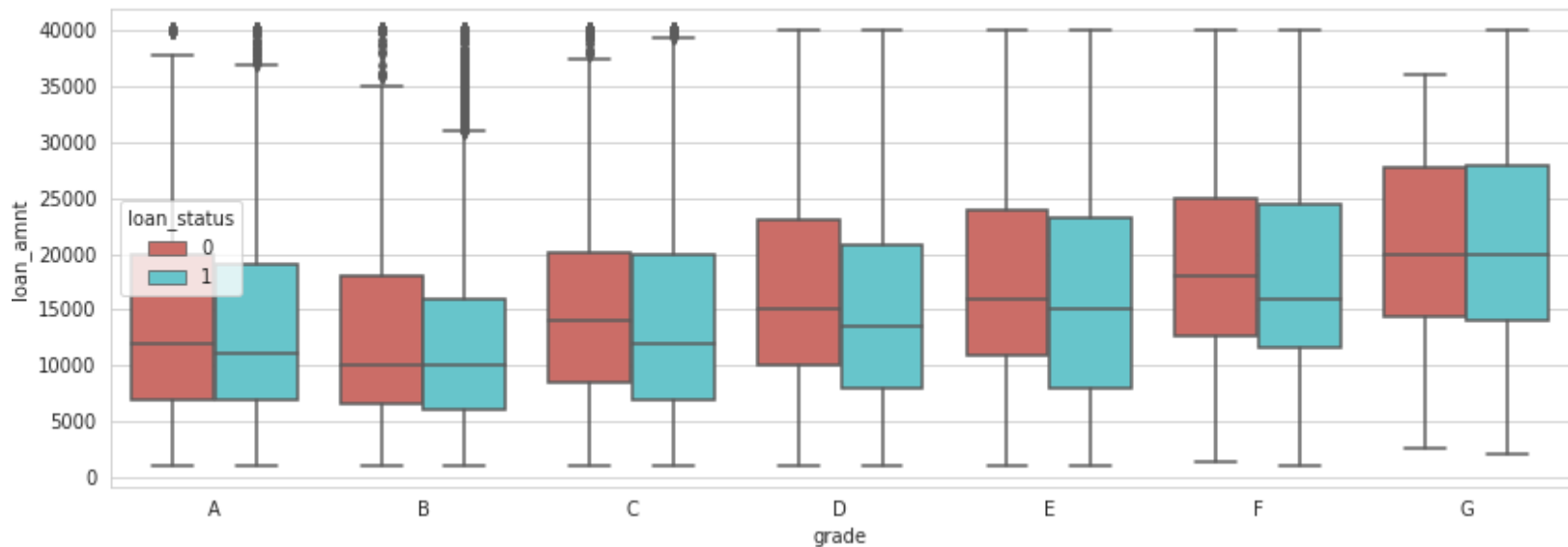


Exploratory Data Analysis



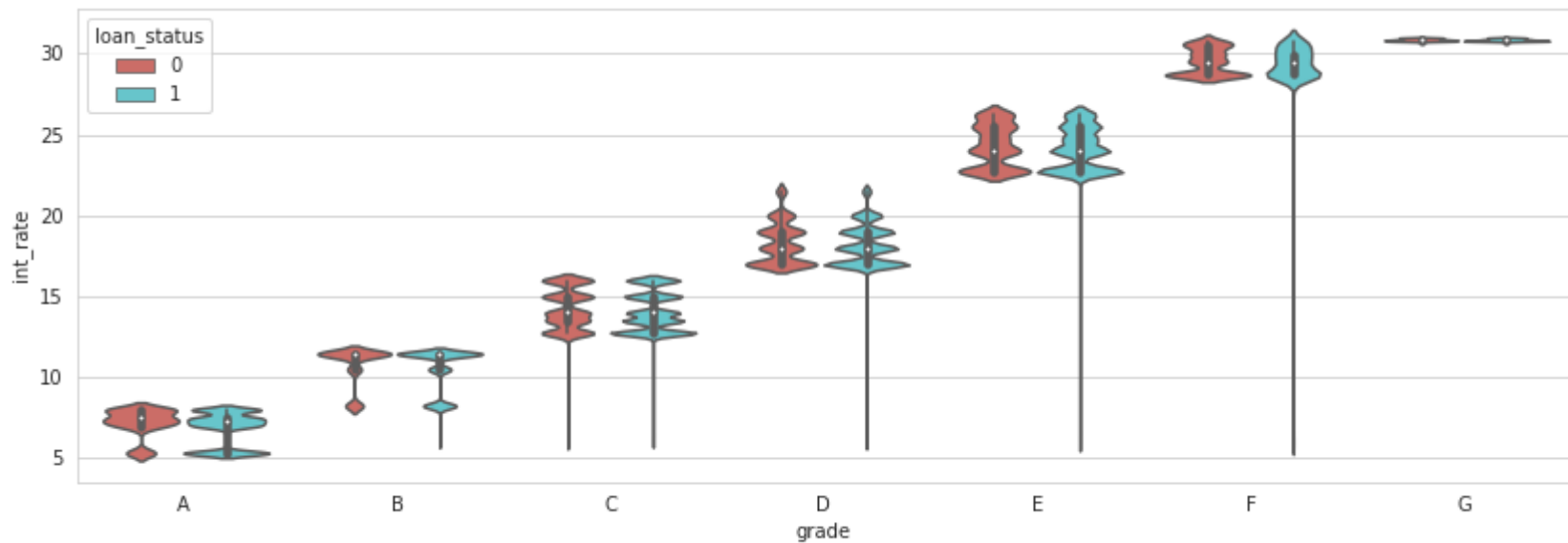
Exploratory Data Analysis

Loan Amount plotted against Credit Grade



Exploratory Data Analysis

Interest Rate plotted against Credit Grade



Further Data Cleaning and Feature Engineering

Mapped the “emp_length” features as below :

```
[25] mapping_dict = {  
    "emp_length": {  
        "10+ years": 10,  
        "9 years": 9,  
        "8 years": 8,  
        "7 years": 7,  
        "6 years": 6,  
        "5 years": 5,  
        "4 years": 4,  
        "3 years": 3,  
        "2 years": 2,  
        "1 year": 1,  
        "< 1 year": 0,  
        "n/a": 0  
    }  
}
```

Further Data Cleaning and Feature Engineering

➤ Removed the following features:

- **Zip Code** : contained 612 discrete values with last 3 digits hidden for each value and I would need to add many dummy variable columns to use it for classification. This would make our Dataframe much larger and could slow down how quickly the code runs.
- **last_credit_pull_d** , **last_pymnt_d** , **issue_d** , **emp_title** : were not significant features.
- **title** : was highly correlated with the feature “**Purpose**”
- **Sub Grade** : was dependent and correlated with feature “**Grade**”

Note :

“**addr_state**” feature could have been dropped too as it contains 49 discrete values and hence 49 dummy variables were created while label encoding this categorical feature but I retained it to avoid missing a could be potential feature/parameter as as employment tenure , income levels , employment opportunities etc. may vary statewise.

Further Data Cleaning and Feature Engineering

- **Applied “*Label Encoding Technique*” to convert the following categorical features in a Machine Learning Model friendly format. All the following are nominal categorical features and none is an ordinal categorical feature:**
 - grade
 - home_ownership
 - verification_status
 - pymnt_plan,
 - purpose
 - addr_state
 - initial_list_status
 - application_type
 - hardship_flag
 - debt_settlement_flag
 - term

Further Data Cleaning and Feature Engineering

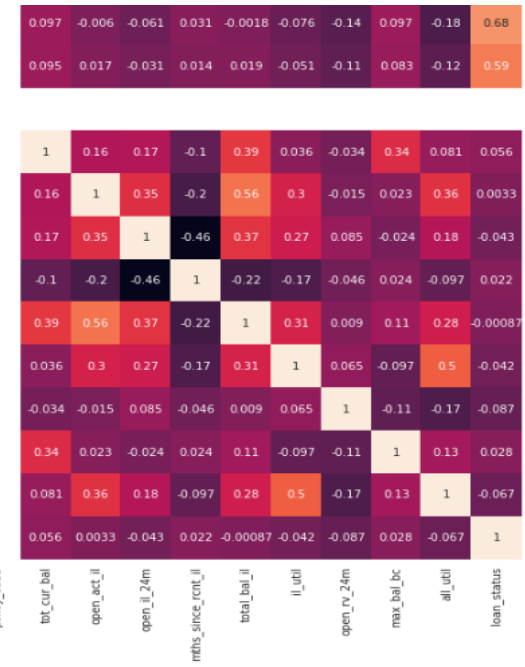
- **Replaced the missing values for following features having missing values with either of their mean , median or mode as appropriate.**
- **emp_length** -> Length of Employment
- **Dti** -> A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income
- **revol_util** -> Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit
- **mths_since_rcnt_il** -> Months since most recent installment accounts opened
- **il_util** -> Ratio of total current balance to high credit/credit limit on all installment accounts
- **all_util** -> Balance to credit limit on all trades
- **bc_open_to_buy** -> Total open to buy on revolving bankcards
- **bc_util** -> Ratio of total current balance to high credit/credit limit for all bankcard accounts
- **mo_sin_old_il_acct** -> Months since oldest bank installment account opened

Further Data Cleaning and Feature Engineering

- **mths_since_recent_bc** -> Months since most recent bankcard account opened
- **mths_since_recent_inq** -> Months since most recent inquiry
- **percent_bc_gt_75** -> Percentage of all bankcard accounts > 75% of limit

Correlation Matrix

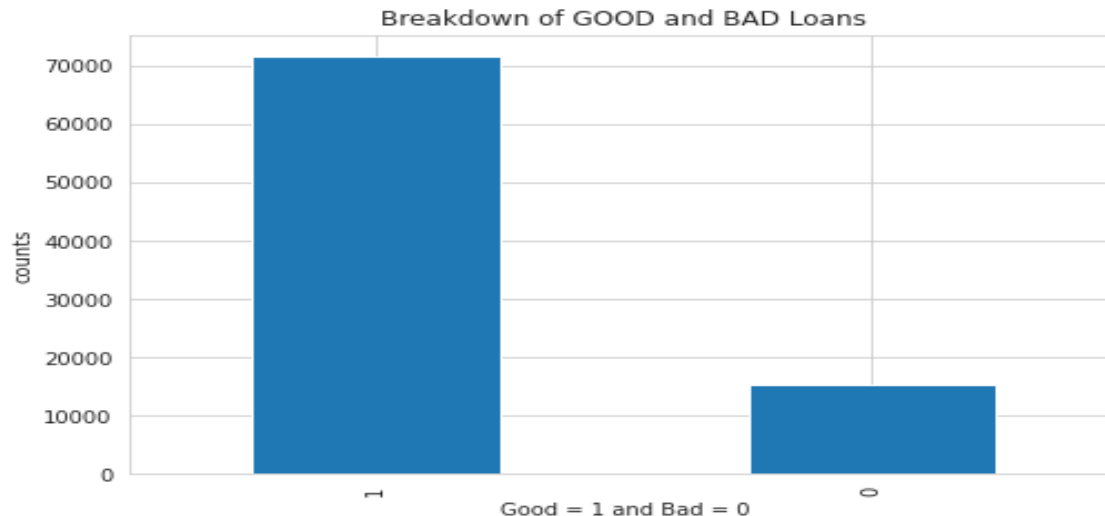
After all the data cleaning , feature engineering and label encoding process, I created correlation heatmaps to show relation of the new 147 features with target variable “**loan_status**” by diving the 147 features in **5 groups** but I didn’t drop any further features based on the correlation heatmaps. In other words , these correlation matrices didn’t add much value.



Exploratory Data Analysis

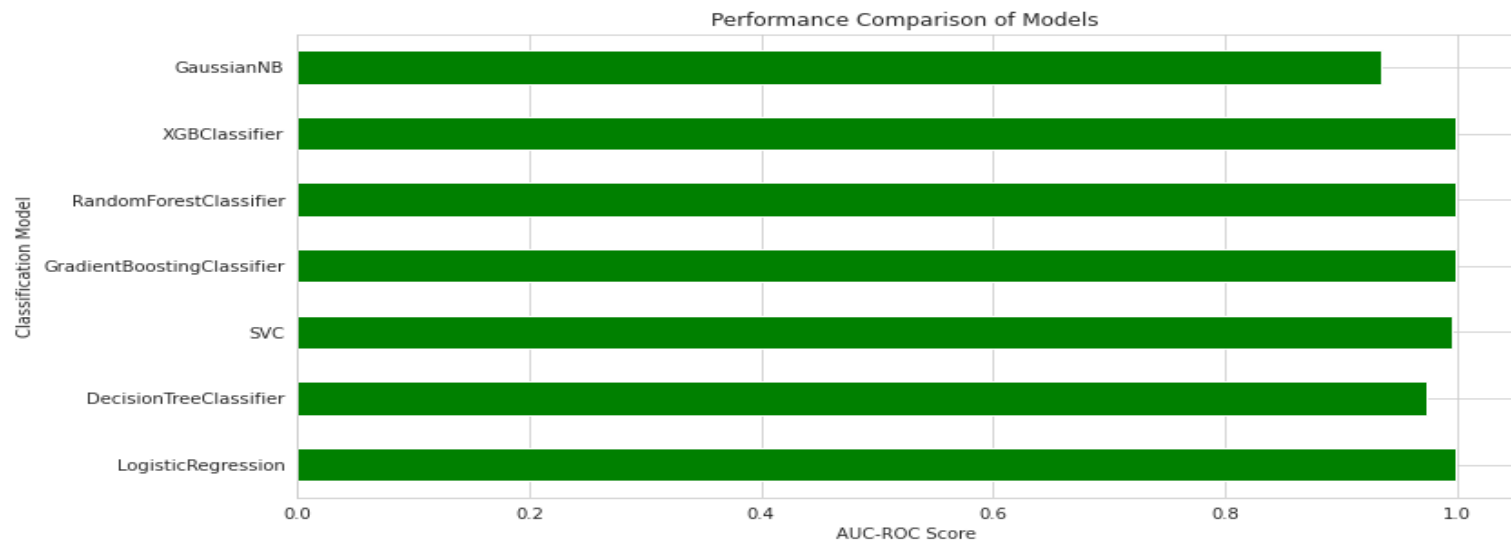
A key feature to highlight before moving onto the modelling section is the distribution of the target class. After the feature engineering was completed, the result was a two-class scenario with bit imbalanced classes as about 17% of the data constituted the minority class (ie Defaulted), the **Stratify** parameter was used during **train_test_split** to ensure that the models were trained accurately to obtain satisfactory results.

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



Model Building

I carried out a rudimentary comparison of a number of different classification models in order to aid the **model selection process**. The scoring I used is the ROC-AUC score, which uses a probabilistic scoring for identifying the ability of the model to capture the different categories of actual outcomes. Logistic Regression , XGBoosting Classifier and Random Forest Classifier produced superior results as compared to other classifiers.

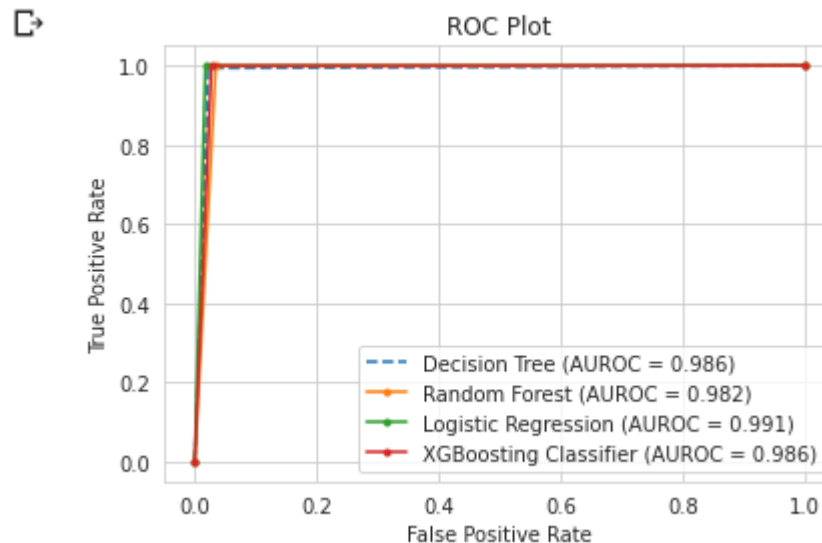


Model Building

I decided to narrow down the focus to four different algorithms. The top three high performing ones , namely , **Logistic Regression, Random Forest, XGBoosting** and the fourth one as **Decision Tree Classifier** to analyze the difference of performance between the top performing three and **Decision Tree Classifier** which was second last as per the performance metrics according to the previous page graph.

Results

The three models, namely , **Logistic Regression** , **XGBoosting** and **Decision Tree** performed fairly well and similarly enough as can be seen from the **ROC-AUC** curve plotted below. This meant that the selection of model could not be made based on the predictive power of the model alone.

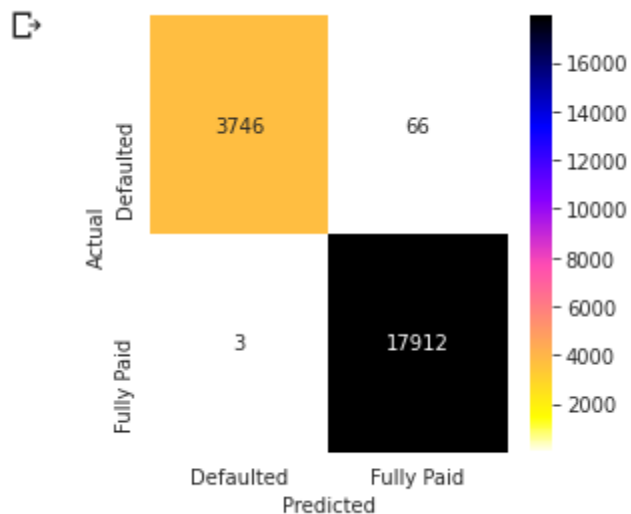


Results

Based on Further Analysis , I selected **Logistic Regression** as the model of my choice based on **Precision , Recall** and **K-Fold Validation Accuracy Scores** with **XGBoosting Classifier** being the second choice of selection.

Logistic Regression :

Confusion Matrix



Results

Precision , Recall and Accuracy Score

```
[87] print(classification_report(Y_test , test_predict_lr , target_names=['Defaulted','Fully Paid'] ))
```

	precision	recall	f1-score	support
Defaulted	1.00	0.98	0.99	3812
Fully Paid	1.00	1.00	1.00	17915
accuracy			1.00	21727
macro avg	1.00	0.99	0.99	21727
weighted avg	1.00	1.00	1.00	21727

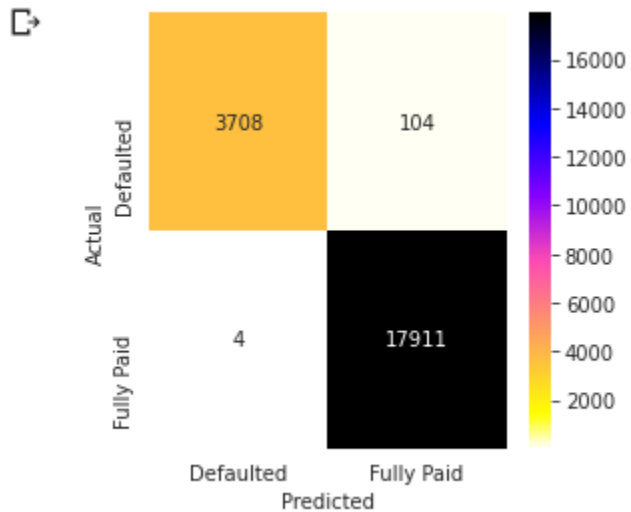
Results

K-Fold Validation Accuracy

Accuracy: 99.72 % Standard Deviation: 0.07 %


XGBoosting Classifier:

Confusion Matrix



Results

Precision , Recall and Accuracy Score

	precision	recall	f1-score	support
Defaulted	1.00	0.97	0.99	3812
Fully Paid	0.99	1.00	1.00	17915
accuracy			1.00	21727
macro avg	1.00	0.99	0.99	21727
weighted avg	1.00	1.00	1.00	21727

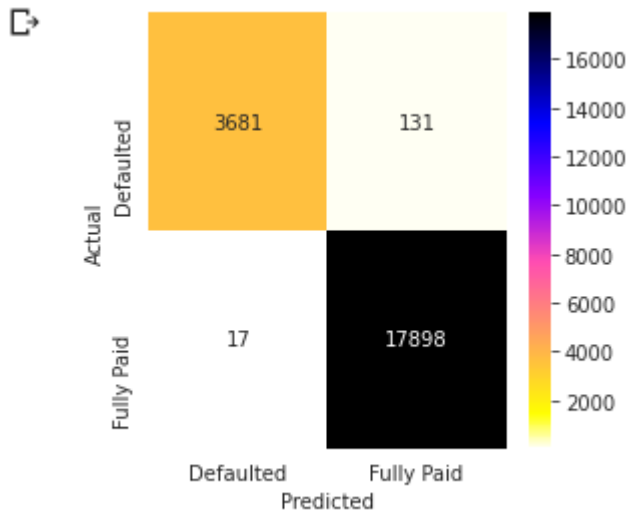
K-Fold Validation Accuracy

Accuracy: 99.55 % Standard Deviation: 0.07 %

Results

Random Forest Classifier:

Confusion Matrix



Results

Precision , Recall and Accuracy Score

	precision	recall	f1-score	support
Defaulted	1.00	0.97	0.98	3812
Fully Paid	0.99	1.00	1.00	17915
accuracy			0.99	21727
macro avg	0.99	0.98	0.99	21727
weighted avg	0.99	0.99	0.99	21727

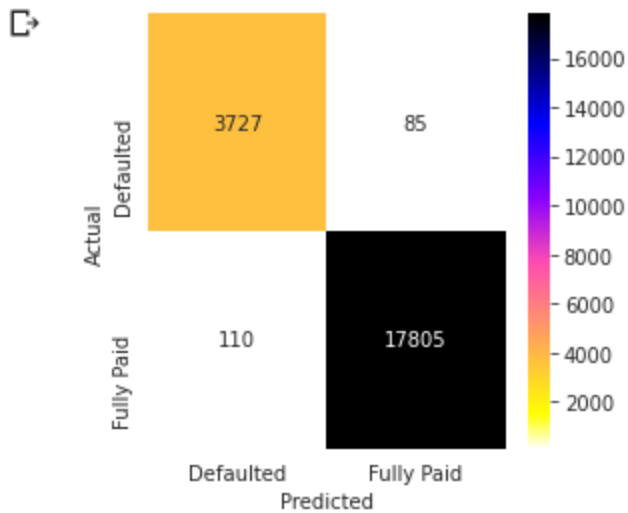
K-Fold Validation Accuracy

Accuracy: 99.36 % Standard Deviation: 0.09 %

Results

Decision Tree Classifier:

Confusion Matrix



Results

Precision , Recall and Accuracy Score

↗	precision	recall	f1-score	support
Defaulted	0.97	0.98	0.97	3812
Fully Paid	1.00	0.99	0.99	17915
accuracy			0.99	21727
macro avg	0.98	0.99	0.98	21727
weighted avg	0.99	0.99	0.99	21727

K-Fold Validation Accuracy

Accuracy: 94.98 % Standard Deviation: 0.21 %

Results

Comparing Logistic Regression with XGBoosting Classifier :

- K-Fold Validation Accuracy(99.72%) for Logistic Regression is higher than K-Fold Accuracy(99.55%) for XGBoosting Classifier and other two models .
- The Precision value for Fully Paid loan status and recall value for Defaulted Loan Status is higher for Logistic Regression as compared to XGBoosting Classifier and other two models
- AUC ROC Curve score is higher for Logistic Regression(0.991) as compared to XGBoosting Classifier(0.986) and other two models

Conclusion

Based on the values of parameters like **Recall** , **Precision** , **Accuracy** , **AUC ROC Curve** and **K-Fold Validation Accuracy Score** , **Logistic Regression** has proved to be the best model to predict *whether a customer will default on Loan or not* followed by the **XGBoosting Classifier**.

Deployment of the best performing model on AWS Cloud

The best performing model Logistic Regression has been re-run with the 9 top important features , namely :

- Loan Amount
- Annual Income
- Total Payment
- Interest Rate
- `revol_util` -> *amount of credit the borrower is using relative to all available revolving credit*
- `last_fico_range_low` -> *The lower boundary range the borrower's last FICO pulled belongs to*
- `num_act_rev_utl` -> *Number of currently active revolving trades*
- `total_bc_limit` -> *Total bankcard high credit/credit limit*
- `total_rev_hi_lim` -> *Total revolving high credit/credit limit*

Deployment of the best performing model on AWS Cloud

The model can be accessed at the following URL of the AWS EC2 instance :

<http://ec2-54-252-26-119.ap-southeast-2.compute.amazonaws.com:8080>

References:

<https://www.kaggle.com/wendykan/lending-club-loan-data>

“Thank You”

