**FINAL REPORT**

**Lending Club Classification using Machine Learning Techniques**

**GROUP - 4**

**Team Members:**

* + - Chirag M Naik
    - Smruthi Srinivas Bhandiwad
    - Chandan Kumar Sinha
    - Kunagu Sai Charan
    - Swati Shrivastava
    - **Mentor:** Mr. Romil Gupta

**CHAPTER 1: INTRODUCTION**

**What is Lending Club?**

Peer to peer lending at Lending Club is a very simple process. It begins with the borrower. They apply for a loan and if they meet certain criteria, their loan is added to Lending Club’s online platform. Investors can browse the loans on the platform and build a portfolio of loans. Each portion of a loan is called a note and smart investors build a portfolio of notes to spread their risk among many borrowers.

* 1. **Objective of the study:**

The Objective of the study is to predict whether the customer applying for a loan through Lending Club will be able to repay the amount or he/she will be delinquent.

* 1. **Need of the study:**

Investors require a more comprehensive assessment of these borrowers than what is presented by Lending Club in order to make a smart business decision, by identifying new borrowers that would likely default on their loans.

* 1. **Business model of the Enterprise:**

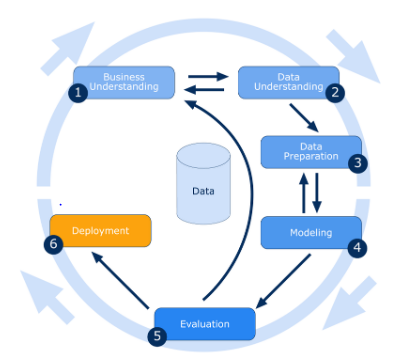
Lending Club is a Peer-to-Peer lending company that utilizes a group of private investors to fund loan requests. Lending Club’s model for risk assessment categorizes borrowers by assigning them a grade and a subgrade based on their credit history.Investors are presented with a list of borrowers, along with their assigned risk assessment grades, and they have the opportunity to choose which borrowers they will fund, and the percentage of funding that they will cover.

Lending Club is an online peer to peer credit marketplace which matches borrowers and investors. For evaluating the credit worthiness of their borrowers, Lending Club primarily relies on a grade and sub-grade it assigns them based on their credit-history. This rating information is then made available to investors who fund the loan requests, so that the investors can decide which loan request and how much of that loan request they will fund. In addition to the grade information, Lending Club provides historical loan performance data to investors for more comprehensive analysis. Our business problem is to find a model that will utilize the historic loan data to help better identify borrowers that are likely to default Such a model would allow investors to avoid loan defaults thus limiting the risk of their investments.

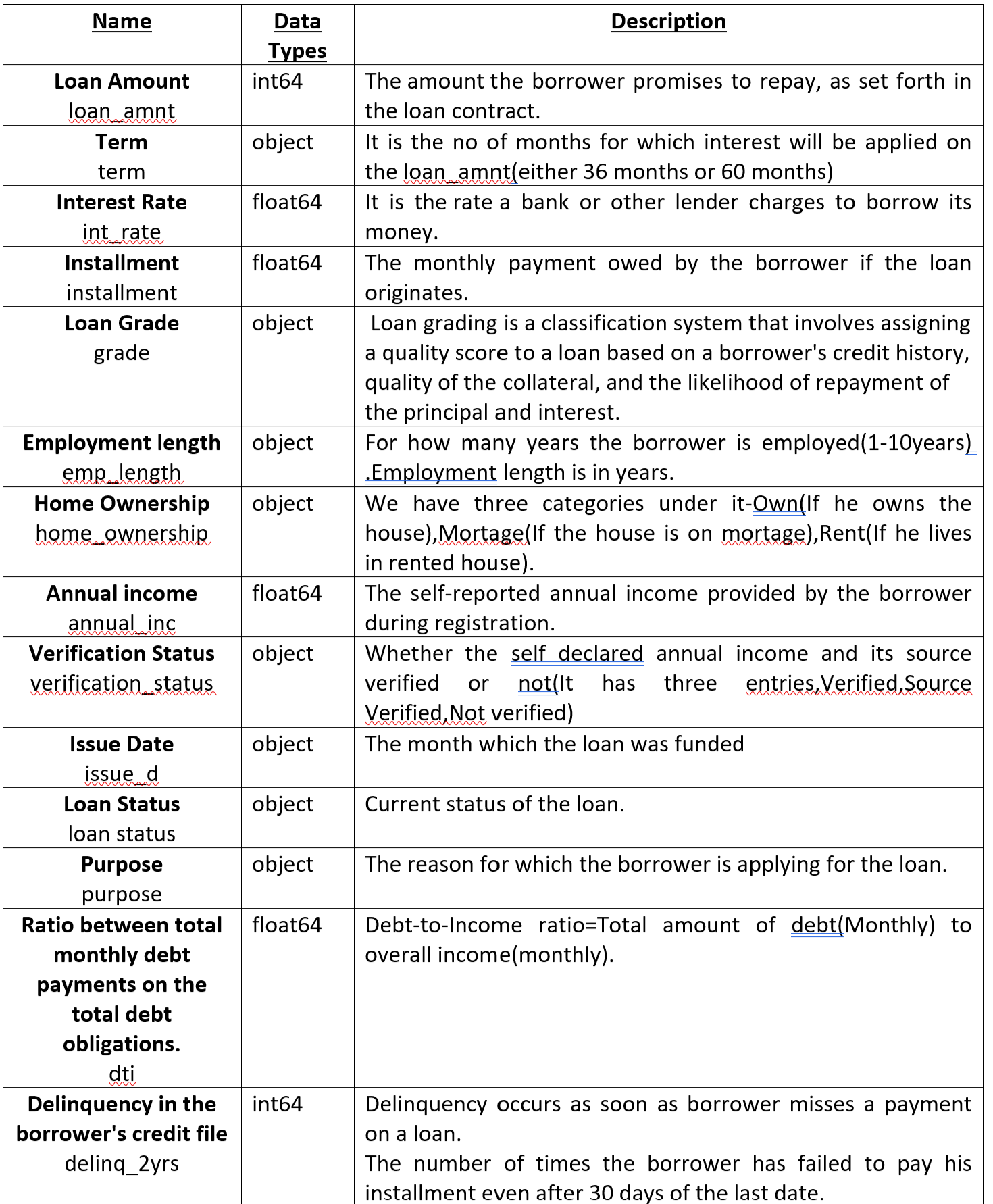
* 1. **Data Source:**

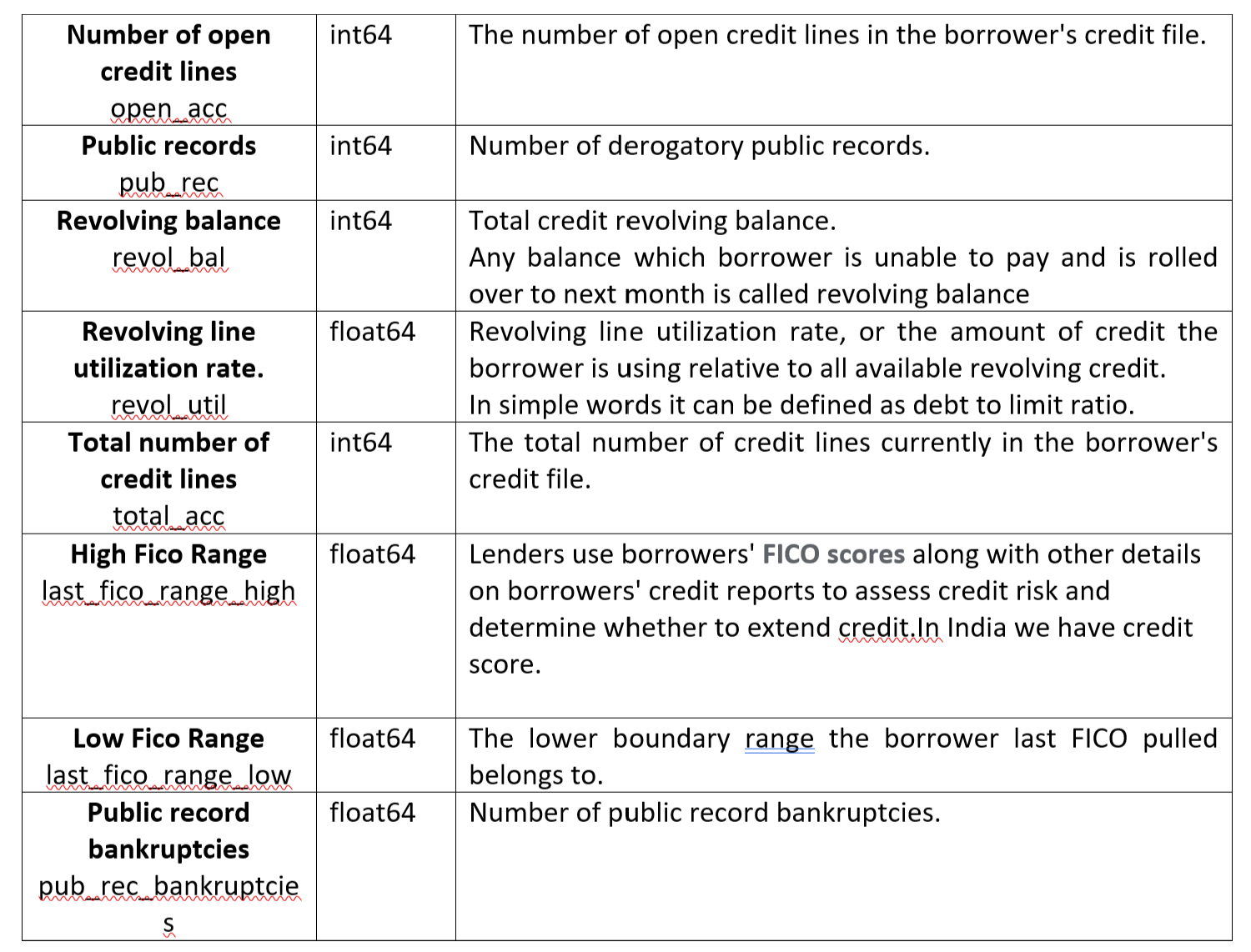
Lending Club provided us with 4 years of historical data ( ). This dataset contained information pertaining to the borrower’s past credit history and Lending Club loan information. The total dataset consisted of over 40,000 records, which was sufficient for our team to conduct analysis models.

**Methodology to be followed:**



**DATA DICTIONARY**

****

****

**CHAPTER 2: DATA CLEANING**

**2.1 Understanding and Interpreting the Data:**

Firstly we imported the basic libraries and then loaded the required dataset. After loading the dataset we checked for the shape of data which is 39786 rows and 150 columns and we checked all the data types of the columns**.**

**2.2 Dropping columns with missing values:**

Firstly we have done the null value treatment. We calculated the percentage of missing values in each column and dropped columns with more than 50% null values.After null value treatment our columns got reduced from 150 to 58.

**2.3 Dropped columns with only one unique category:**

Now we dropped all the columns which have only one unique category as it wouldn’t provide any insight to our model.

Columns have now reduced to 46.

**2.4 Removing features whose info isn't useful for our model.**

We dropped those features which are little to no use for our model.  
Now number of columns have reduced to 23.

Dropped ‘fico\_range\_low’ and ‘fico\_range\_high’ columns and made one column i.e.fico average and two more columns got reduced from the data set.

Checked the relation between target variable and ‘addr\_state’ and removed ‘addr\_state’ feature because it has little significance. ‘purpose’ and ‘title’ give the same info so have to drop one which is less burden to our model.So we dropped‘title’ feature.Finally, our columns got reduced from 58 to just 20 columns which we will be using for training the model.

**2.5 Conversion of Data type**

Removed the percentage symbol and converted into a numeric value from object data type.

After looking at the values of features pub\_rec\_bankruptcies,pub\_rec,delinq\_2yrs it has only two states.So it is converted into object.

**2.6 Missing values imputation:**

After doing the above mentioned steps we were left with three columns that still had some missing values.

* The “**pub\_rec\_bankruptcies**” and “**revol\_util**” column were imputed using mode.
* Rows with null values in “**emp\_length**” were dropped as no visible pattern was there which could have served as a rationale for null value imputation.

Finally the data is cleaned which we will be using for model building and we have saved the cleaned data as “cleaned\_data.csv”.

**CHAPTER 3: DATA PREPROCESSING AND EDA**

**3.1 Understanding the new dataset.**

We have imported all the required libraries and then read our new cleaned file which we have saved as “cleaned\_data.csv”.

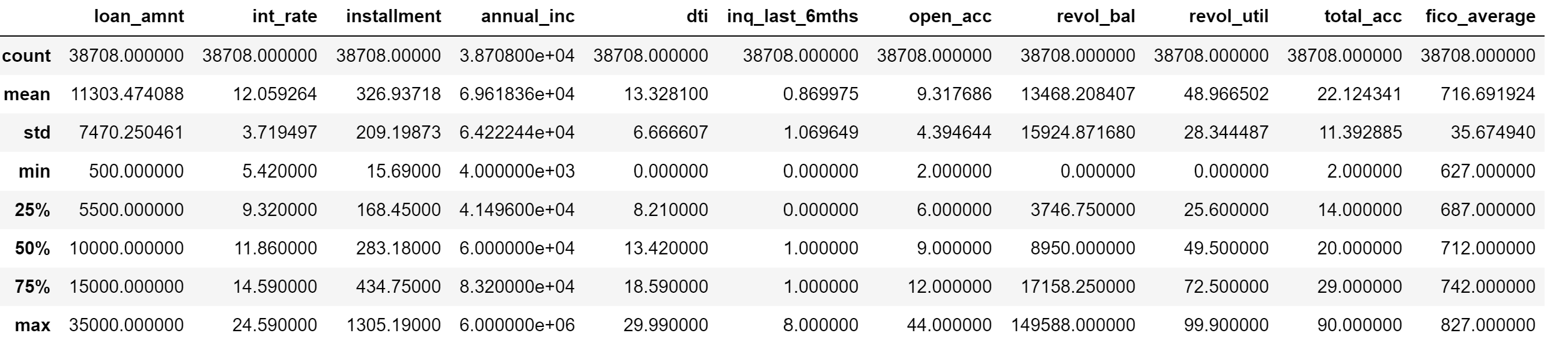
By using shape, info and dtypes function we checked basic for basic information of dataset.

After this we have checked for numerical features in the data set.

**Continuous Variables :-**

'loan\_amnt', 'installment', 'annual\_inc', 'dti', 'delinq\_2yrs','inq\_last\_6mths', 'open\_acc', 'revol\_bal', 'revol\_util','total\_acc', 'fico\_average'.

**3.2 Descriptive Statistics:**

****

###### **Inferences**:

1) The Loan amount column min is 500 and max is 35000, while the mean is 11303 and SD is 7470, the distribution is not normal and is skewed towards left . the bank has more customers with smaller loan amounts.

2) The Installment column min is 15.69 and max is 1305, while the mean is 326 and SD is 209, the distribution is not normal and is skewed towards left . the bank has more customers with smaller loan amounts. this shows similar trend as loan amount

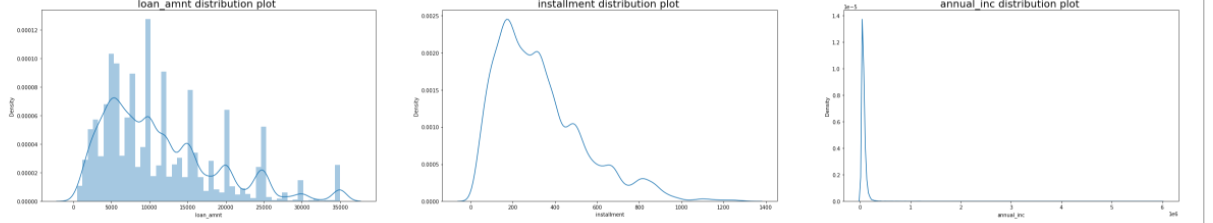
3) The dti ratio column min is 0 and max is 30, the mean is 13.32 with SD of 6.66 is nearly normal distribution

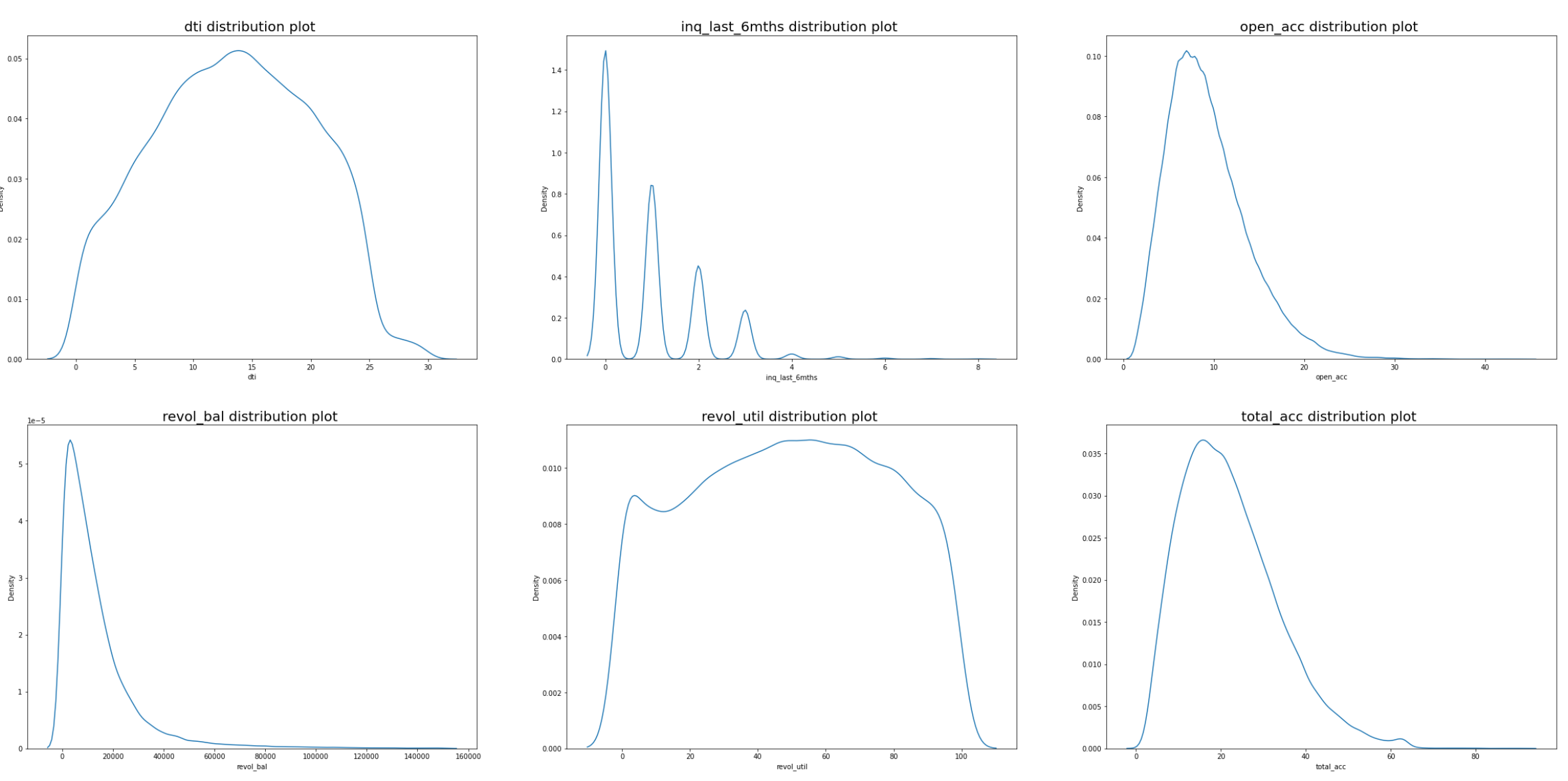
4) The delinq\_2yrs column with min 0 and max of 11 is highly skewed

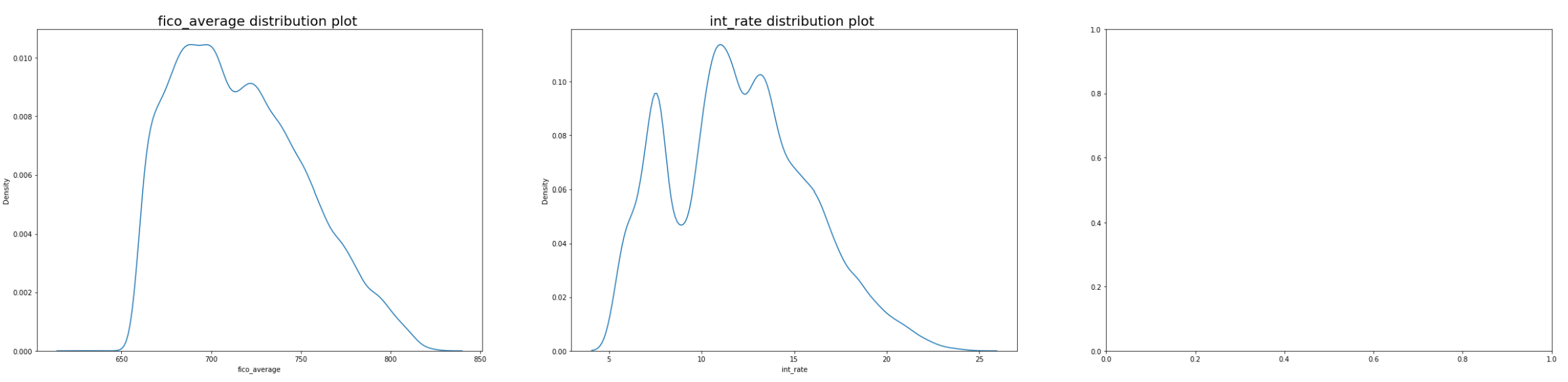
**3.3 Data Analysis:**

**3.3.1 Univariate Analysis:**

By creating distribution plot for each numerical variable, we did univariate analysis:

****

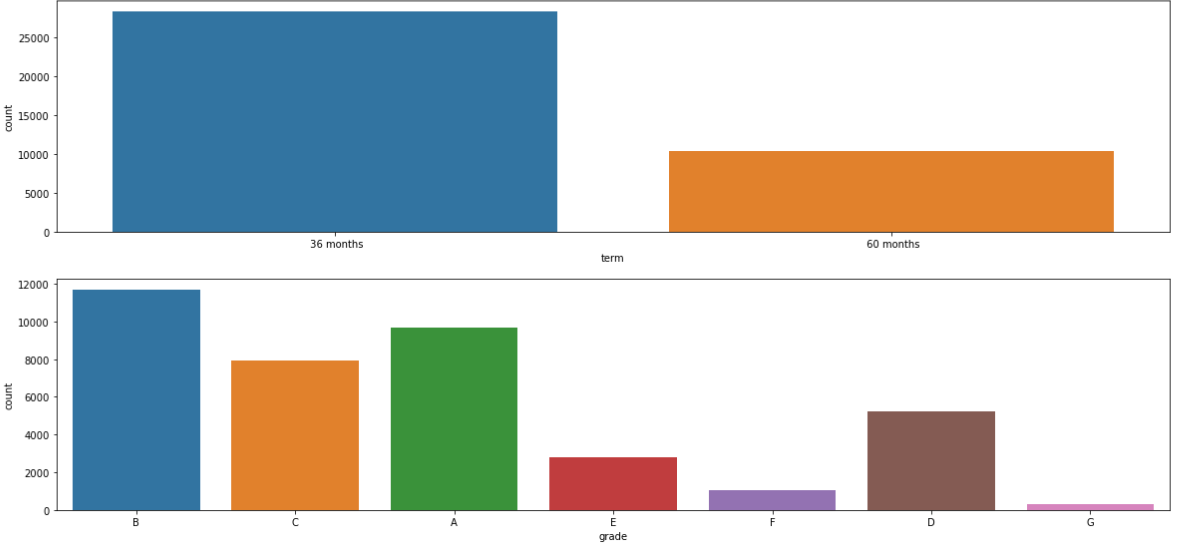
****

****

**Below are the categorical variables** :- 'term','grade','emp\_length','home\_ownership','verification\_status','purpose’

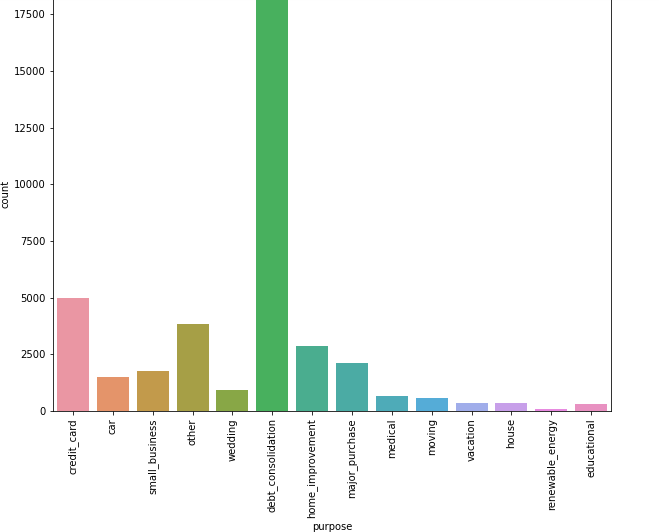
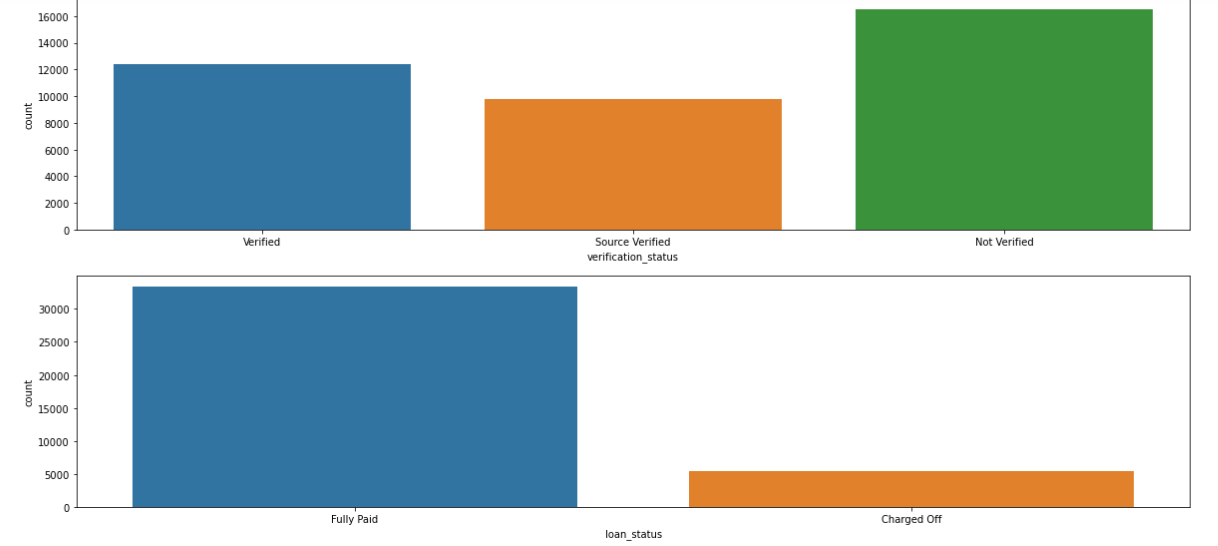
'loan\_status','delinq\_2yrs', 'pub\_rec','pub\_rec\_bankruptcies’.

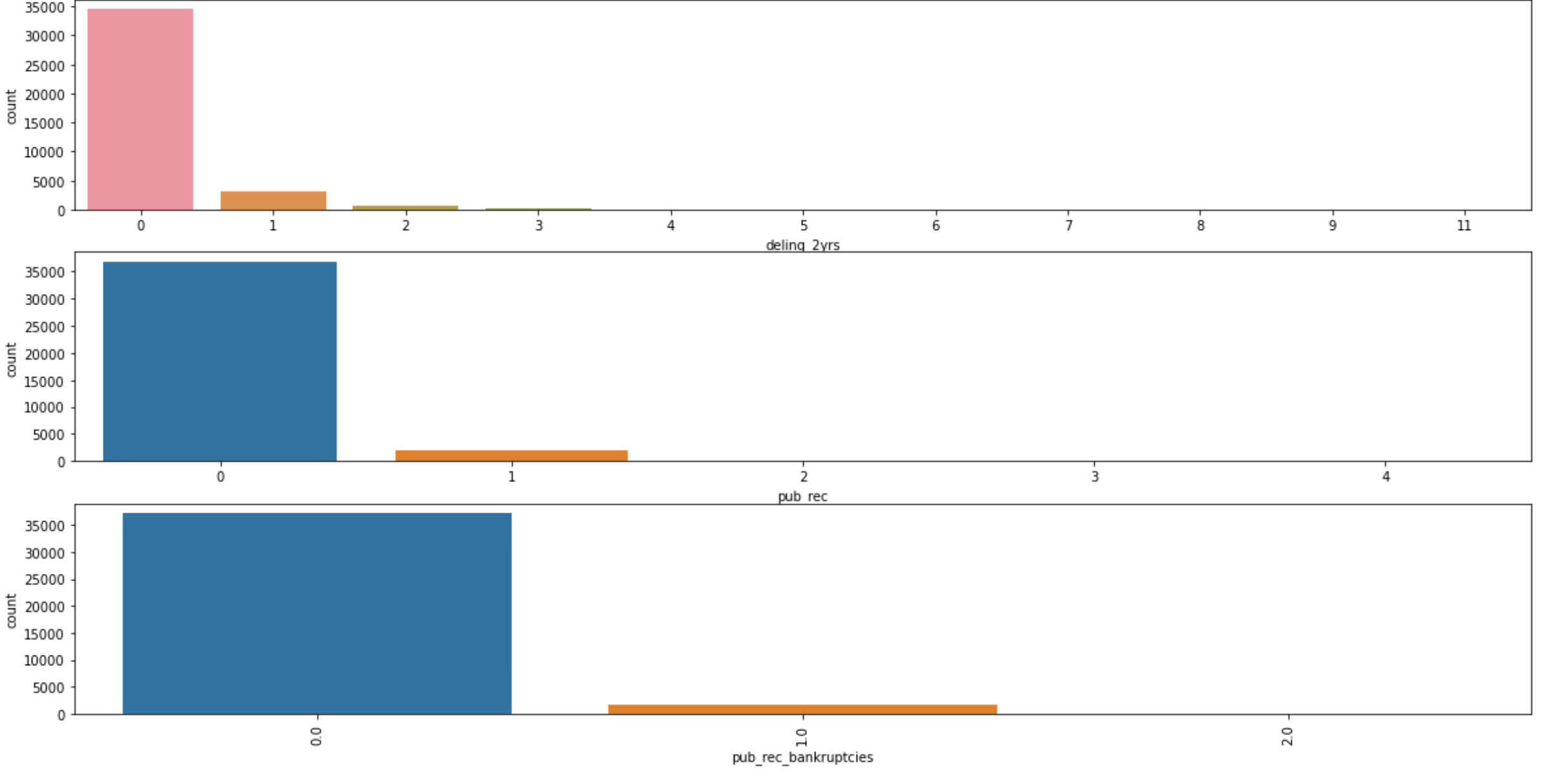
**By creating count plot for each categorical variable, we did univariate analysis:**

****

**Chart, bar chart

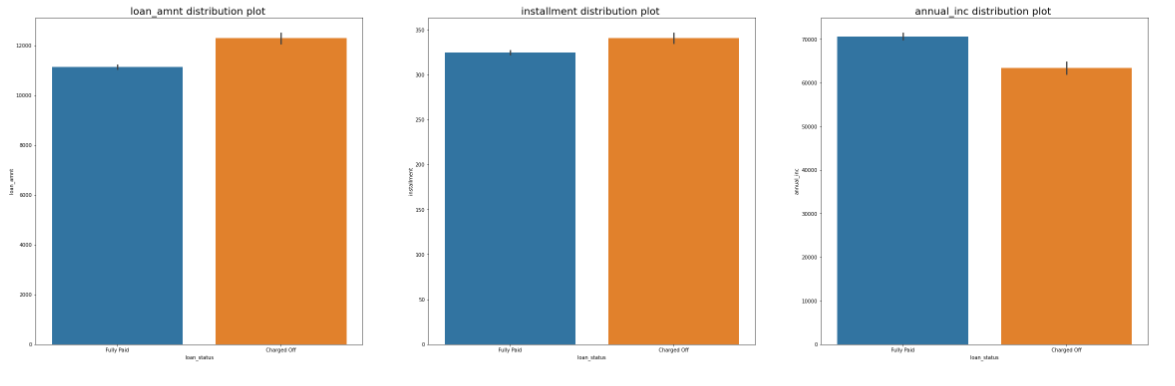
Description automatically generated**

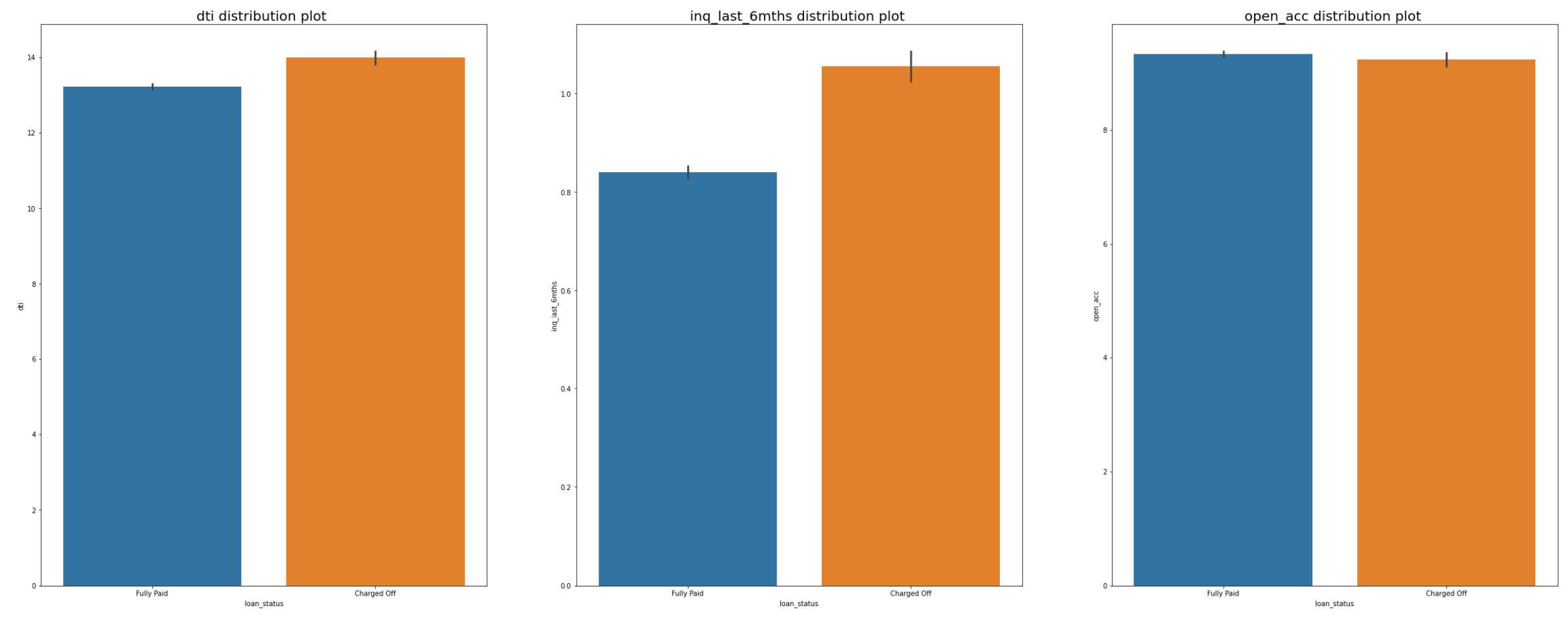
****

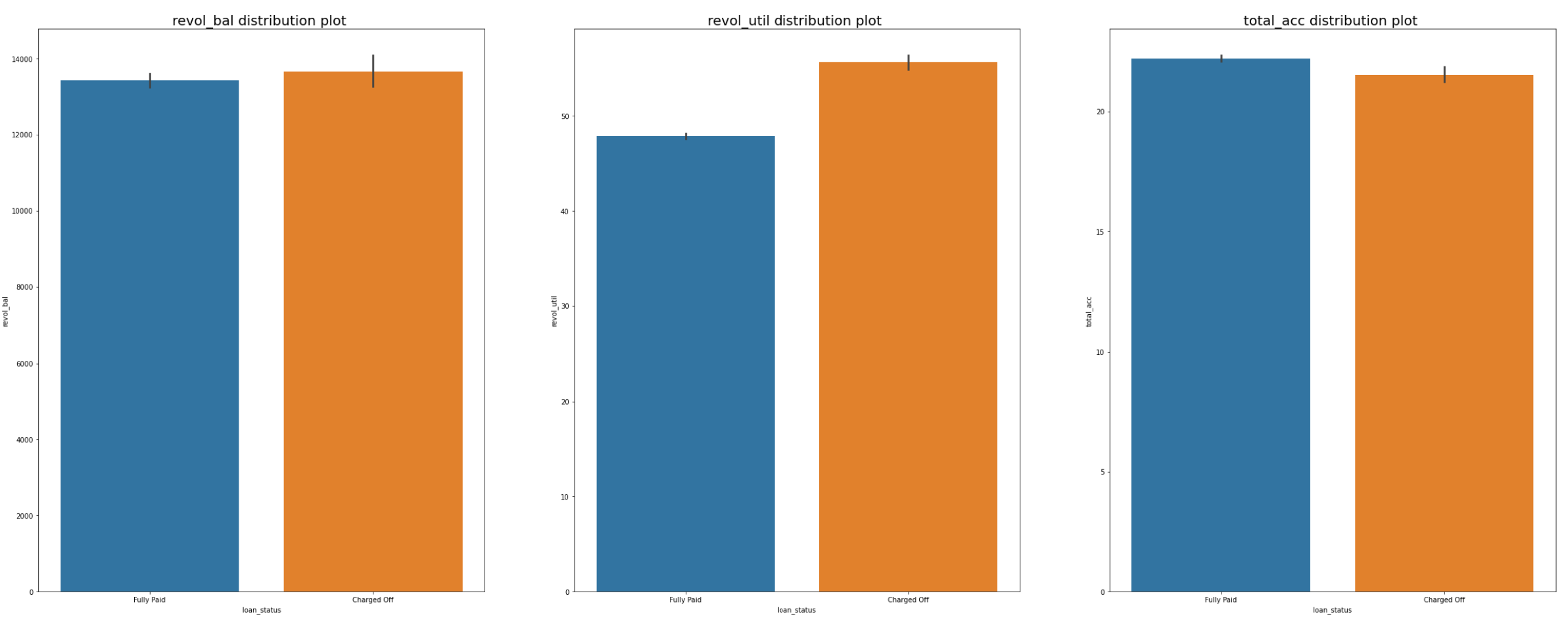
****

**3.3.2 Bivariate Analysis:**

We did bivariate analysis on each numerical variable with target variable by using barplot.

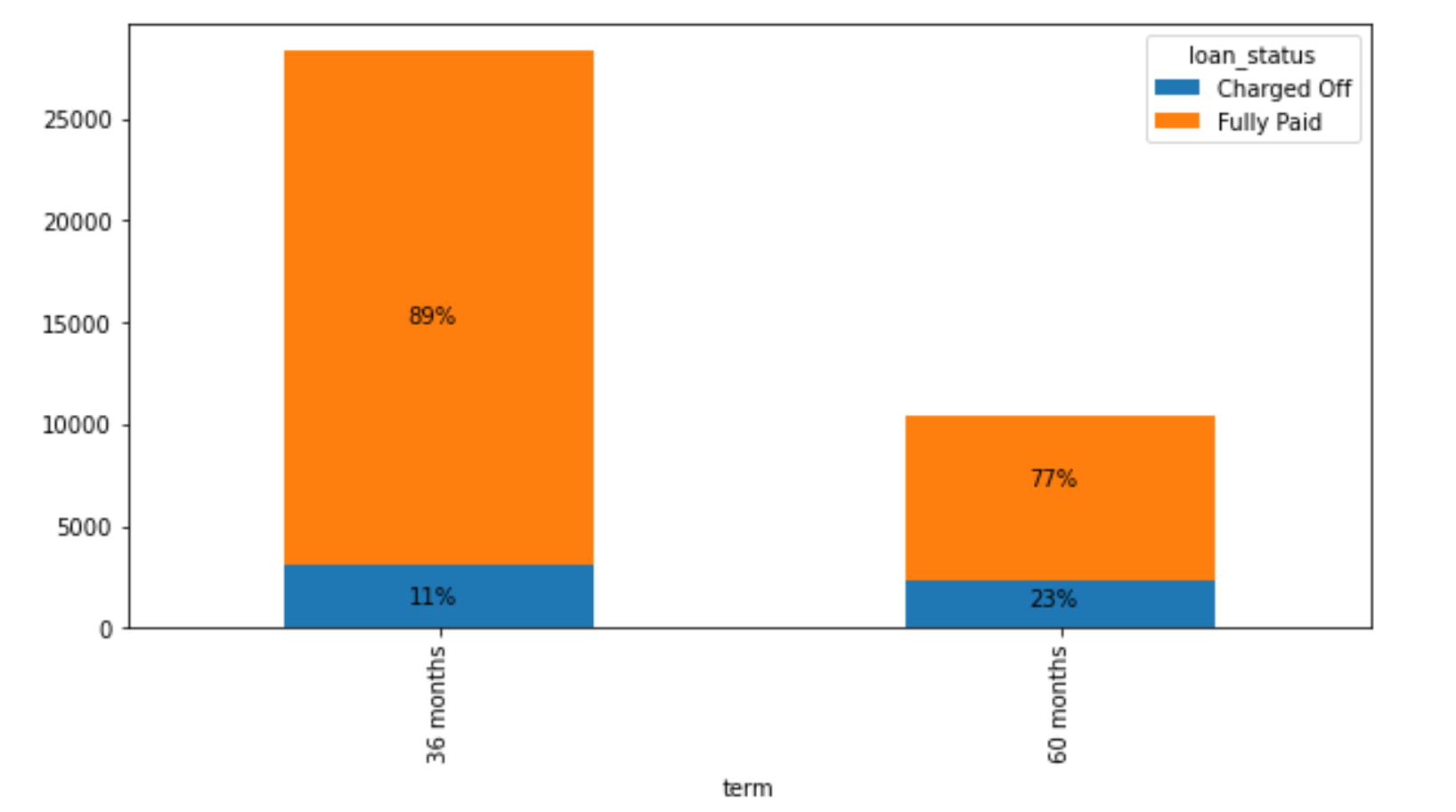


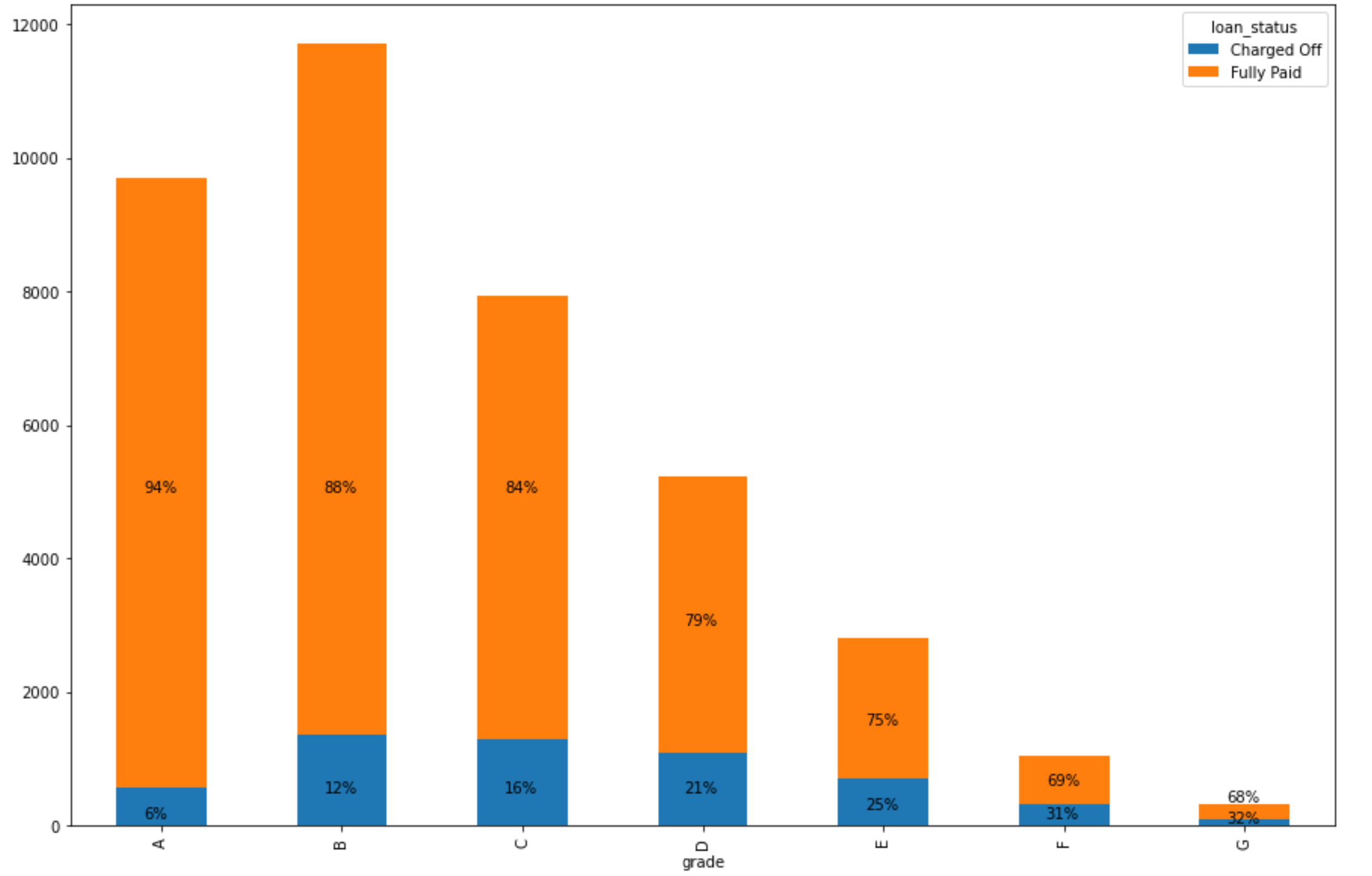


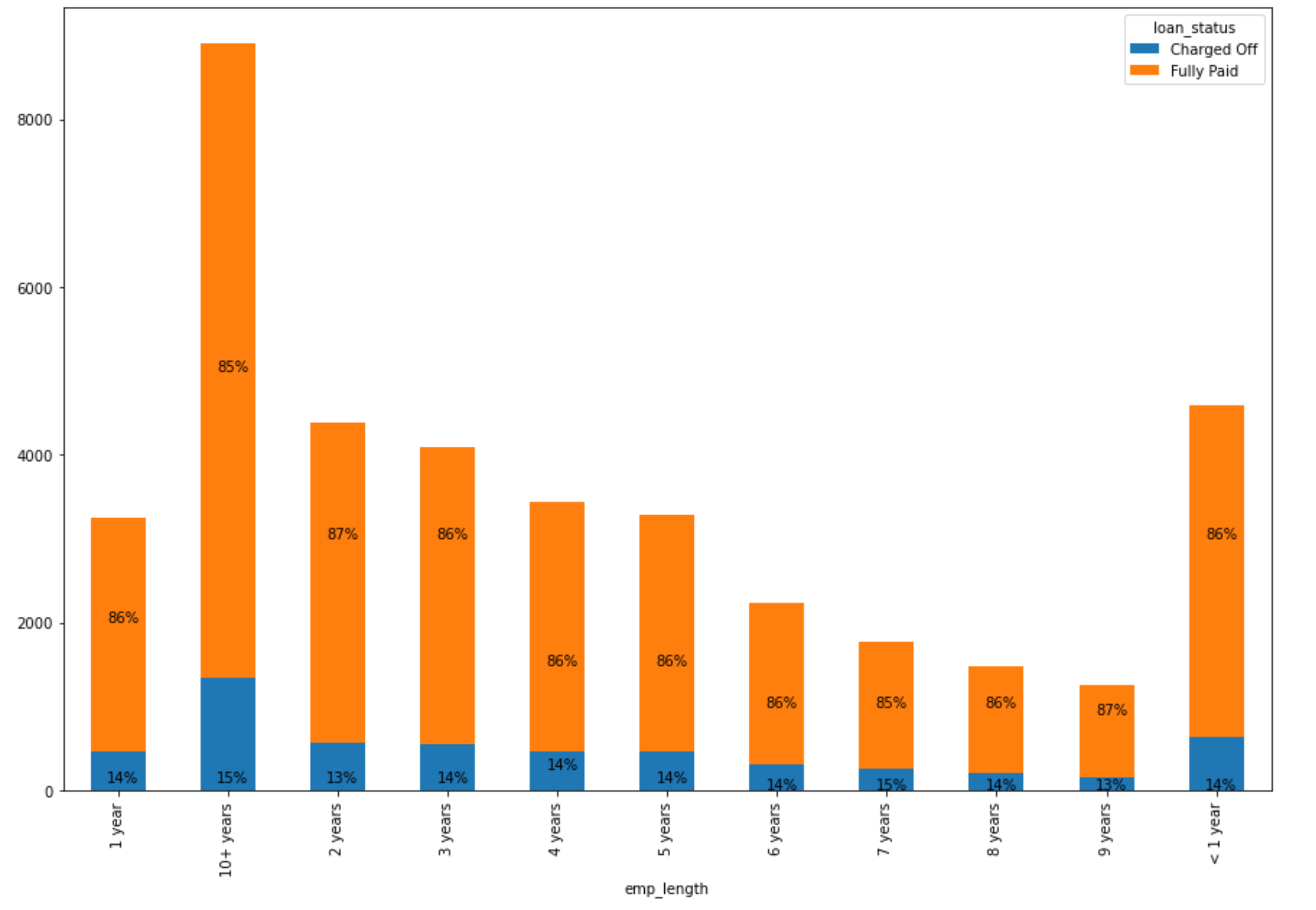


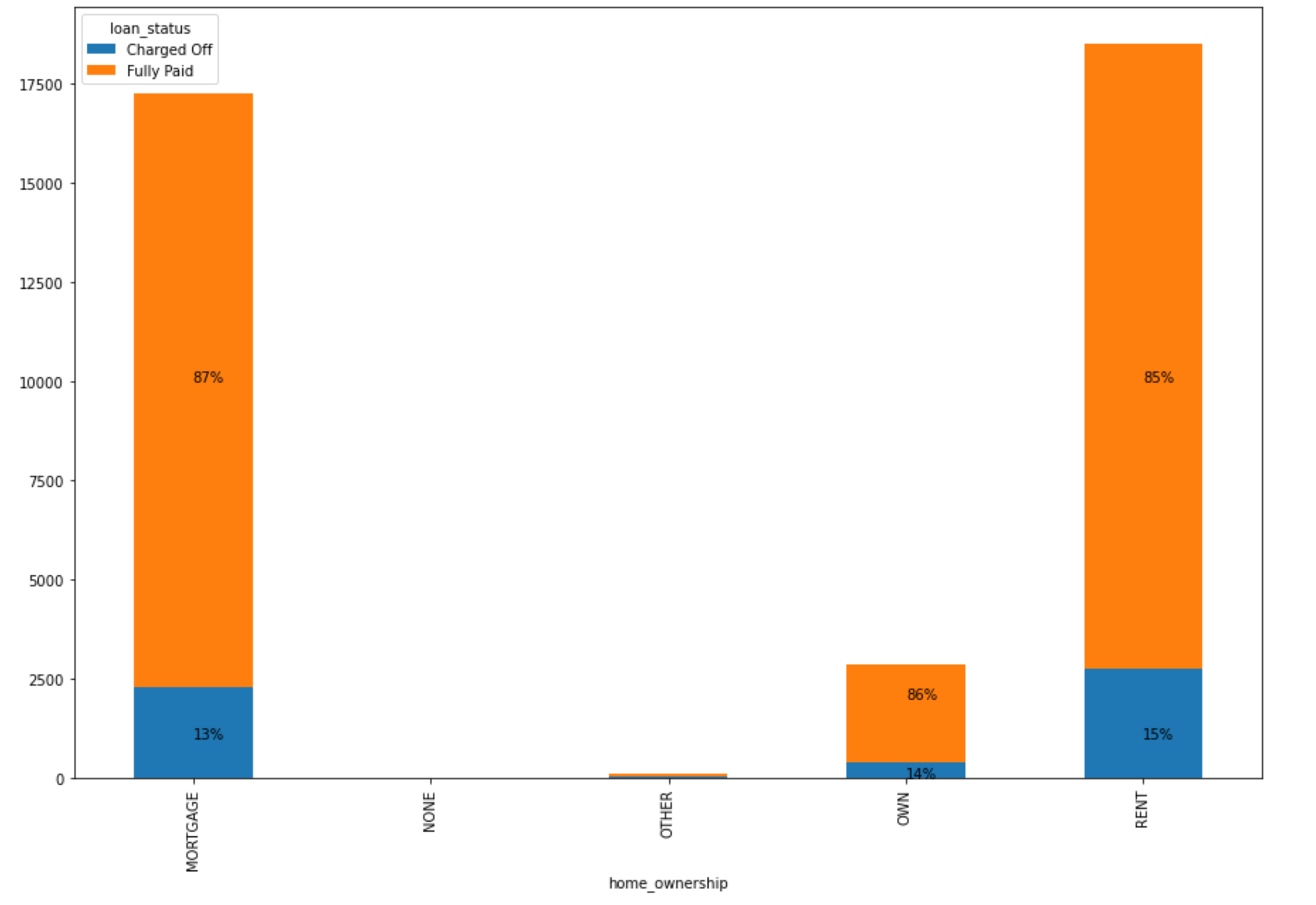


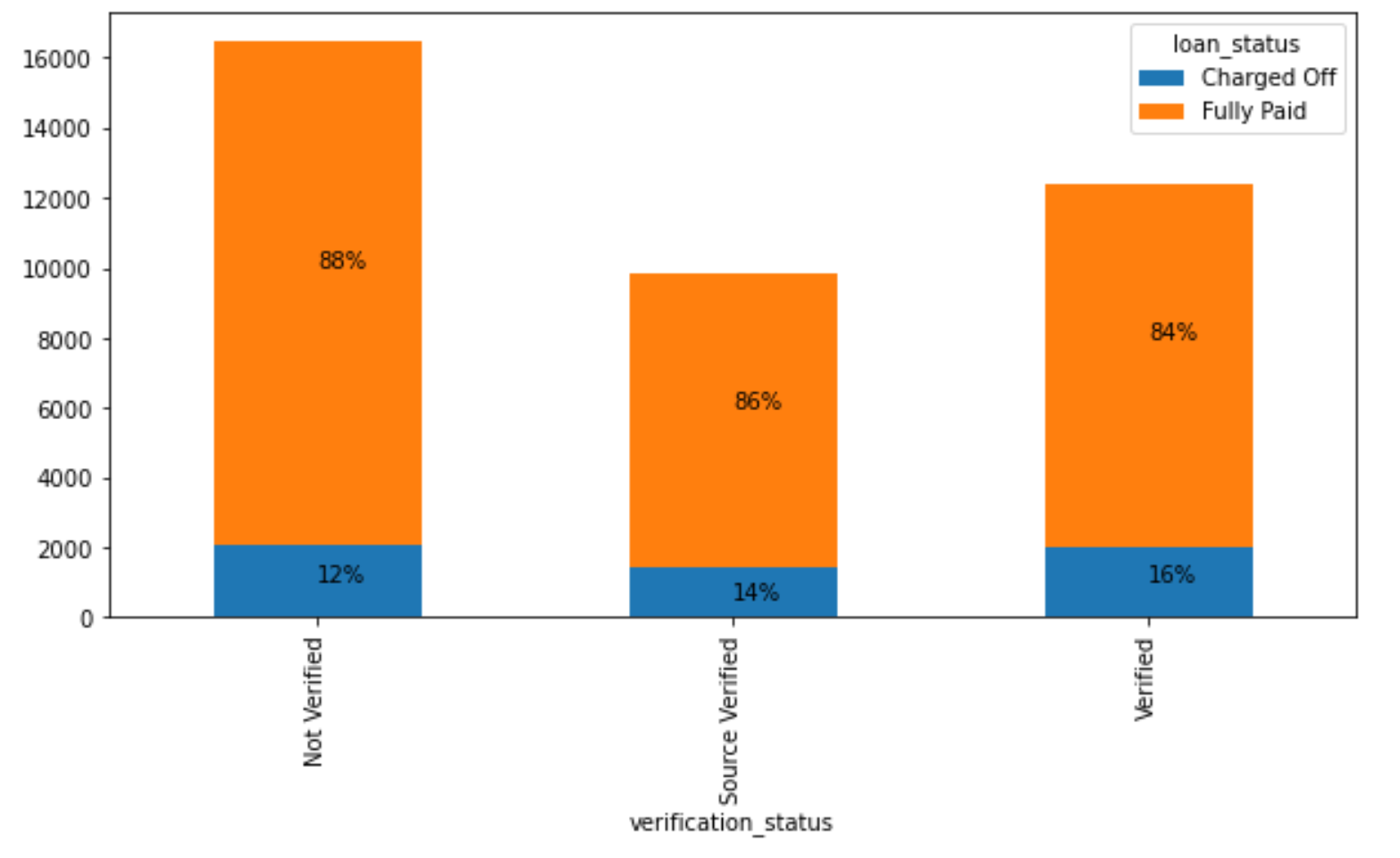
*We performed bivariate analysis for categorical variables with the target variable loan status using cross tab and stacked bar and found the percentage of each categorical variable with respect to the target variable.*[*¶*](http://localhost:8888/notebooks/Downloads/EDA%20(1).ipynb#Inference:-we-have-performed-bivariate-analysis-for-categorical-variables-with-the-target-variable-loan-status-using-cross-tab-and-stacked-bar-and-found-the-percentage-of-each-categorical-variable-with-respect-to-the-target-variable.)

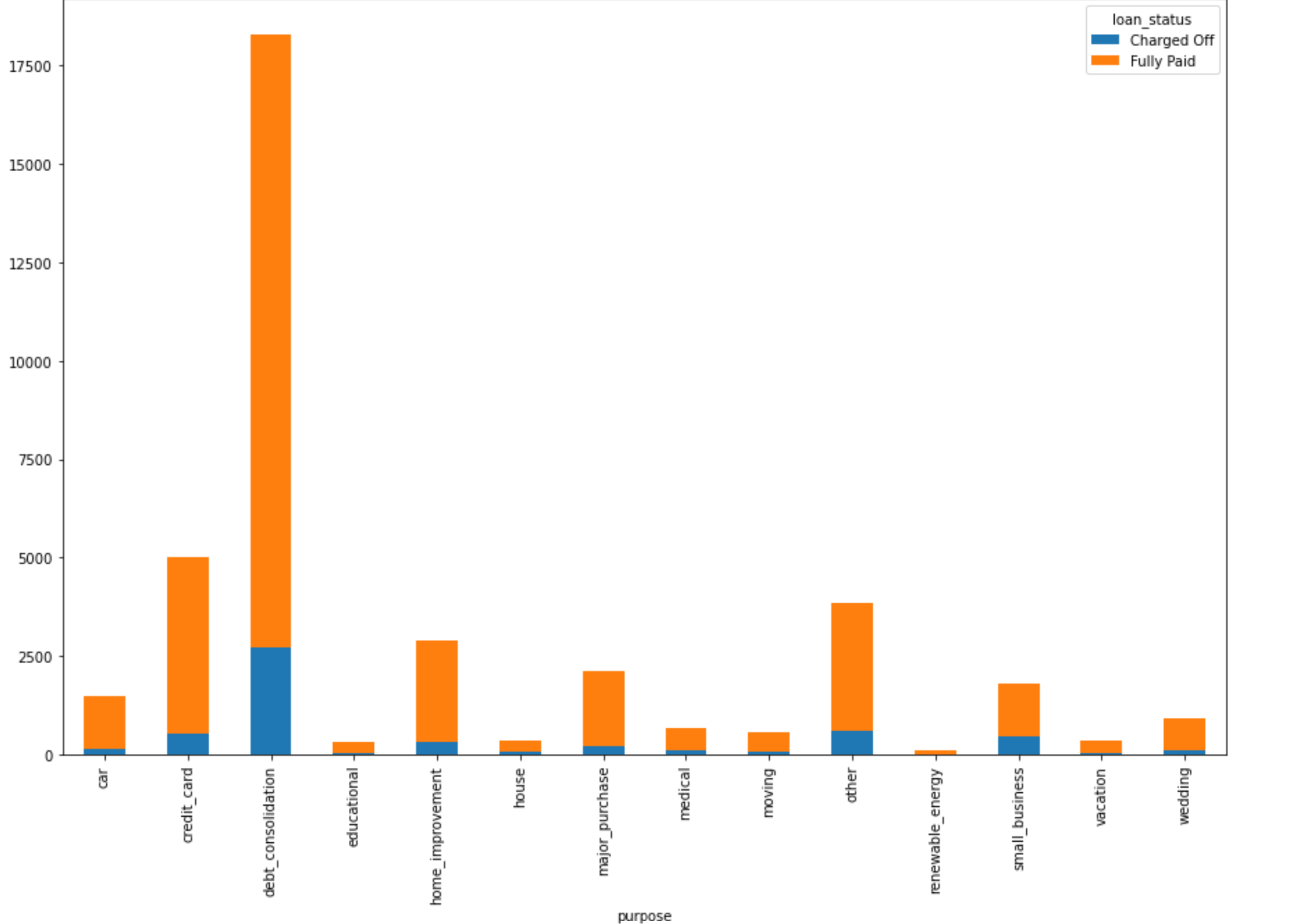
****

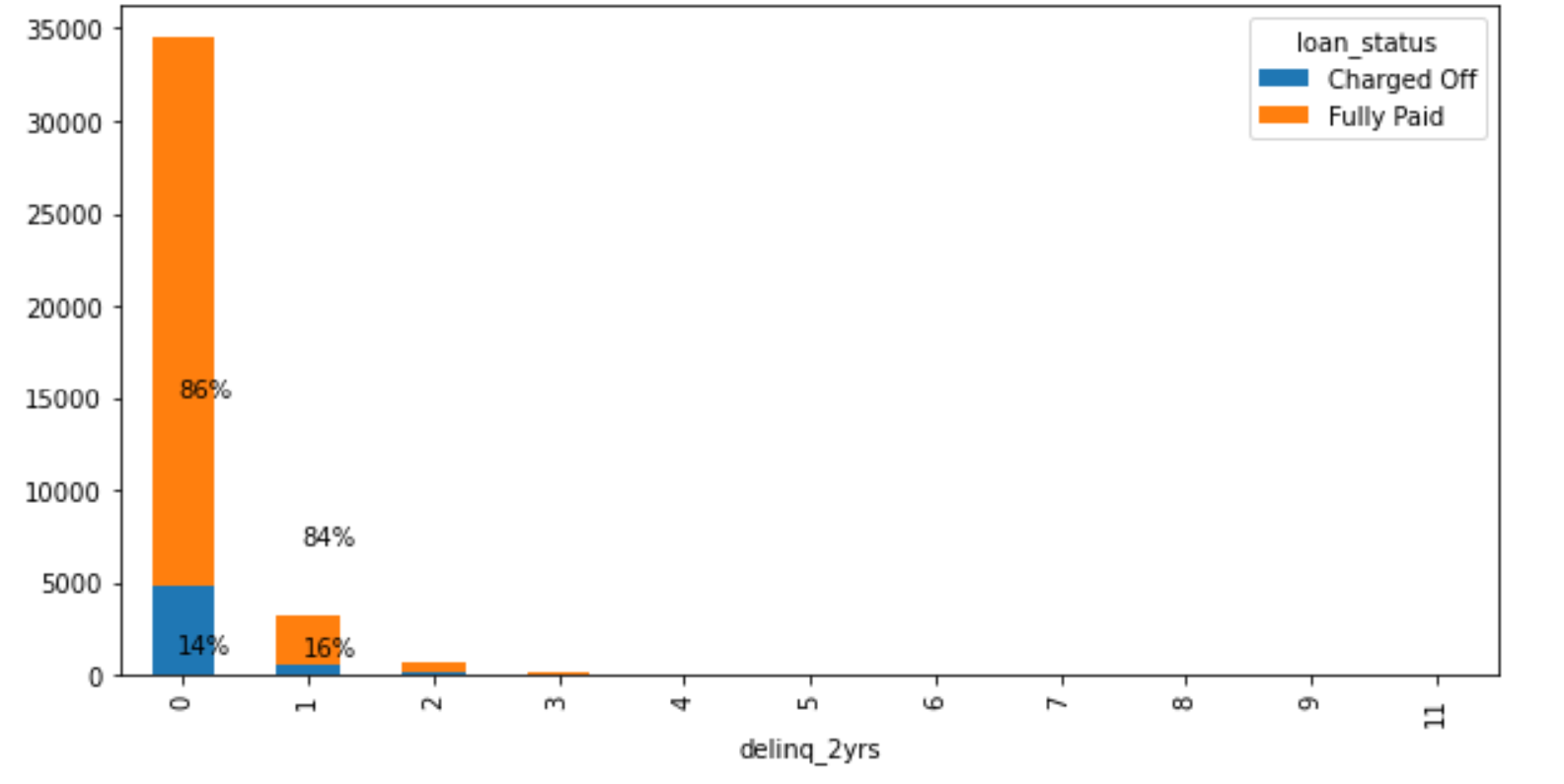
****

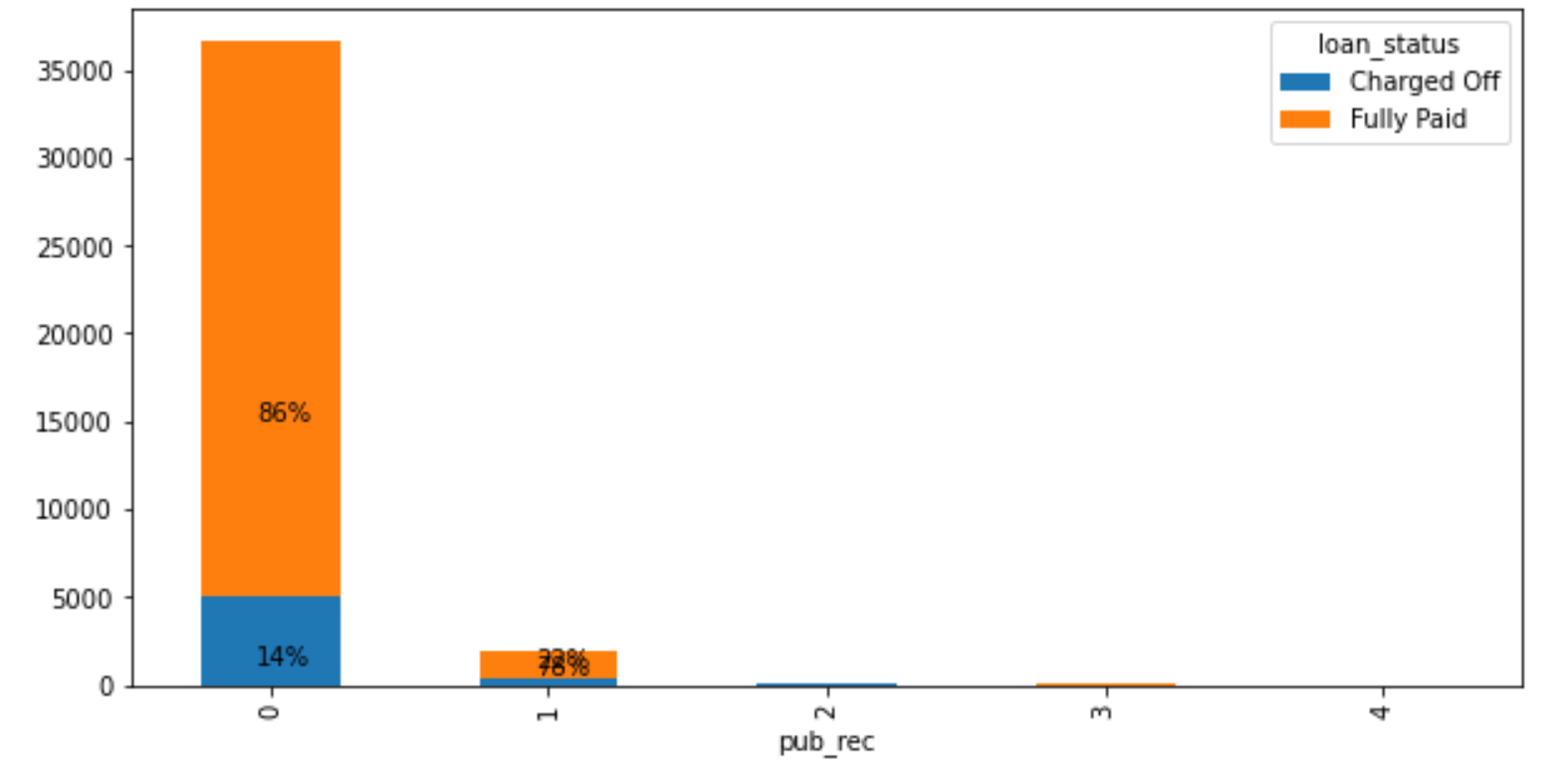
****

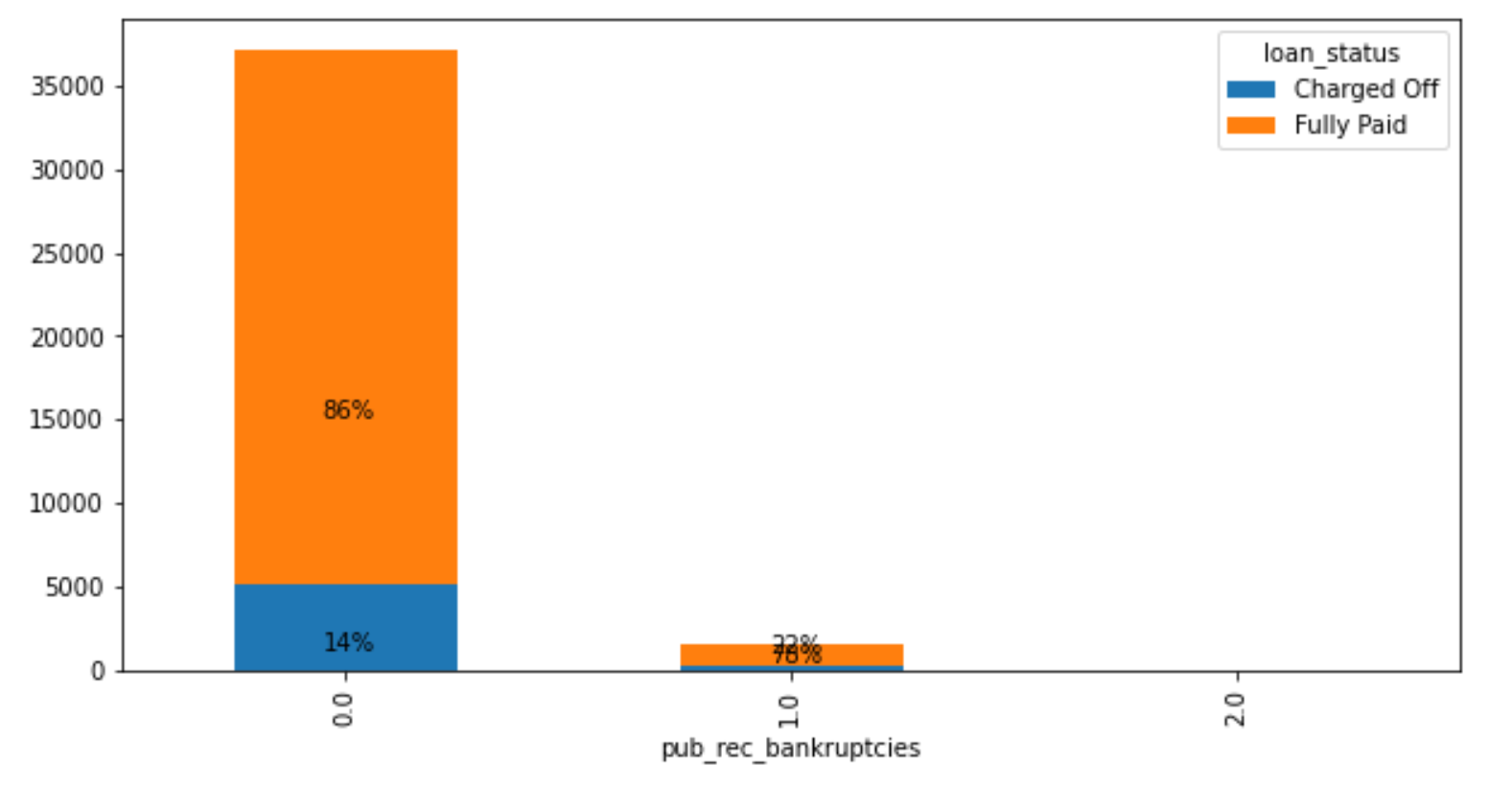
****

****

****

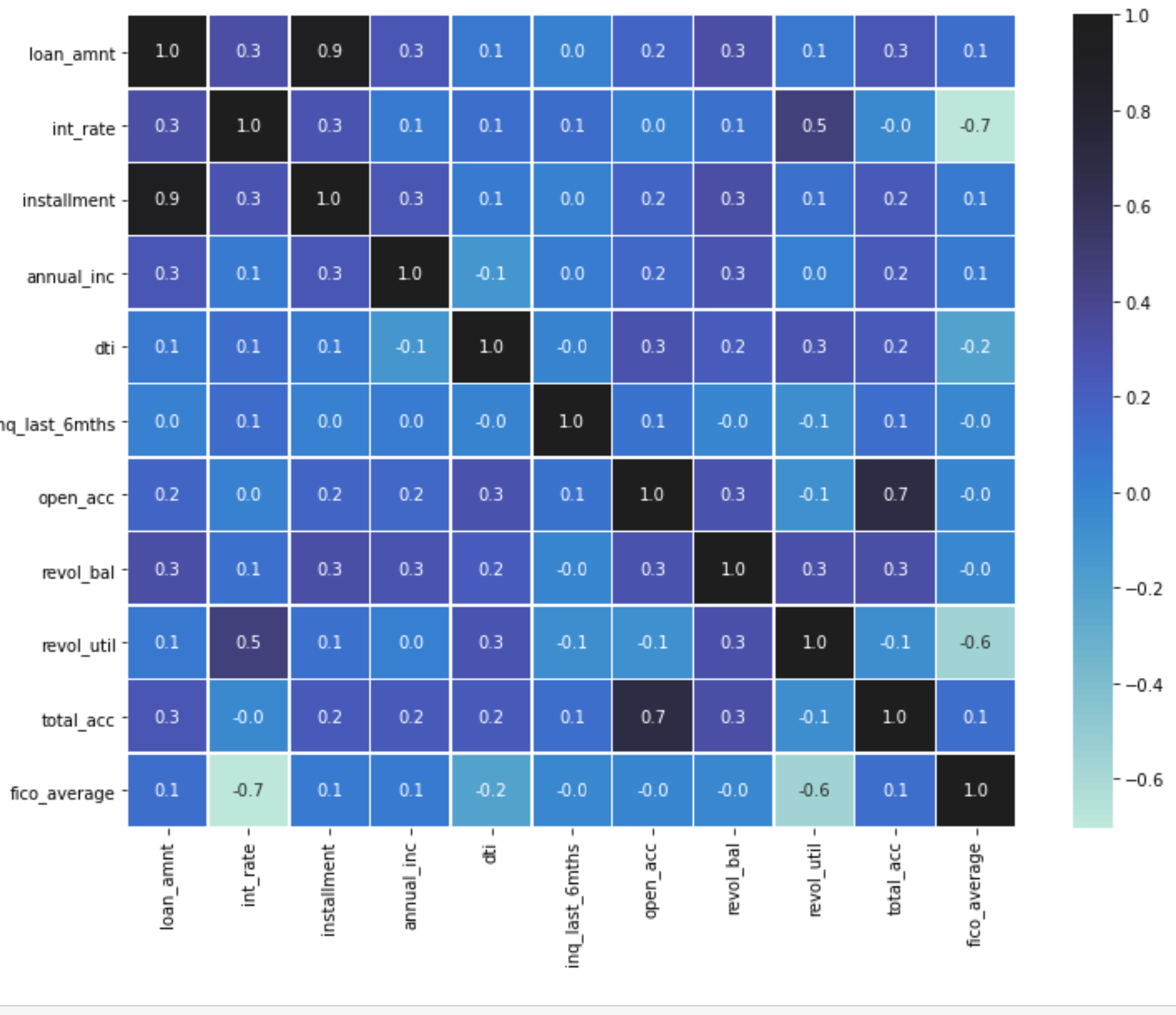
****

****

****

**3.3.3 Multivariate Analysis:**

By using corr() function we did multivariate analysis.



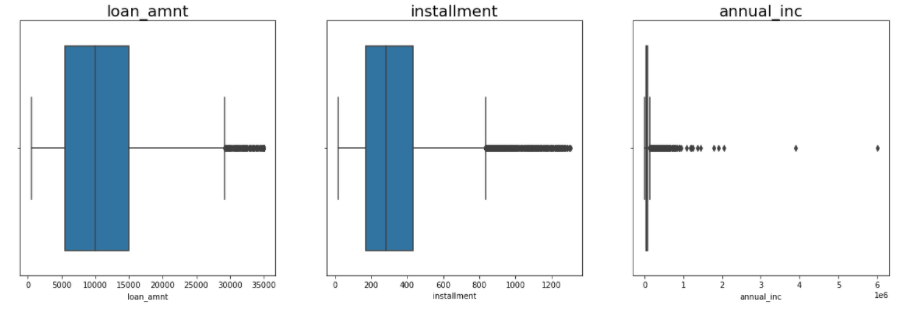
# df.corr()

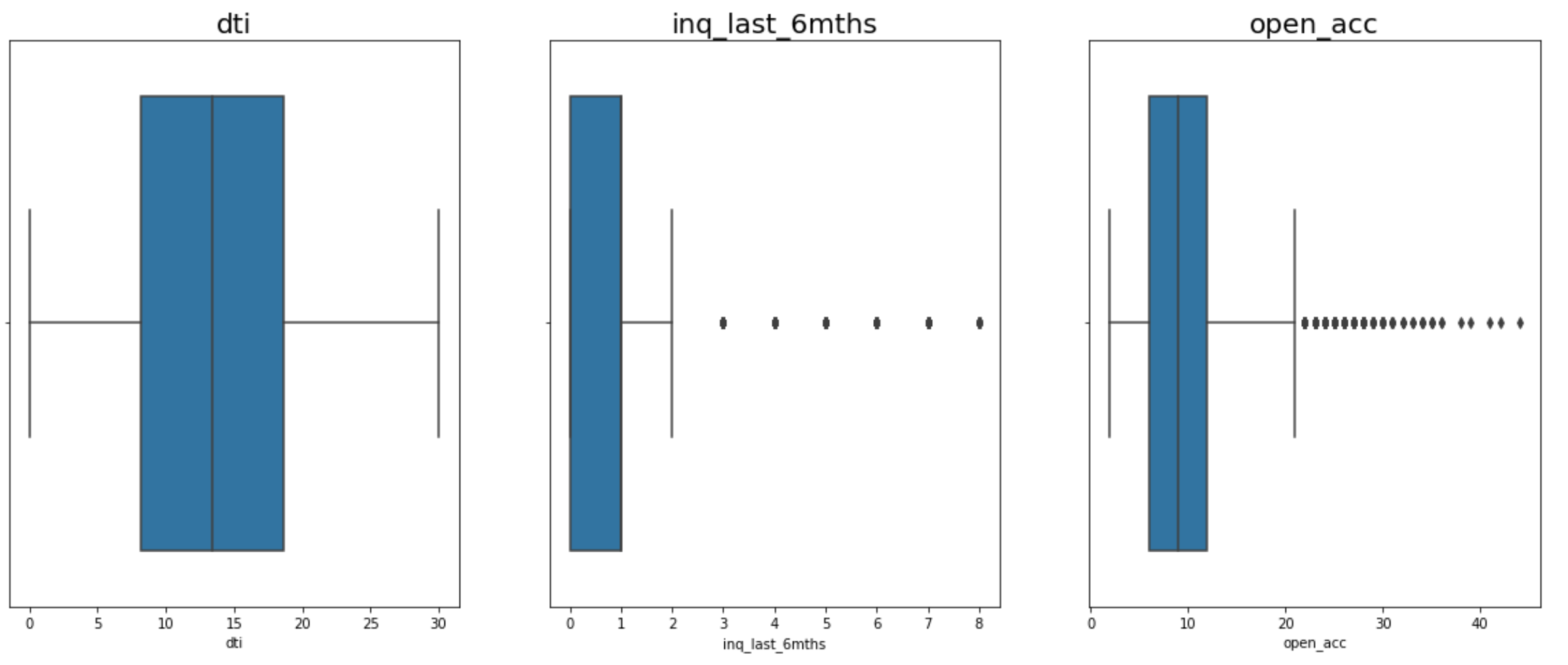
# 

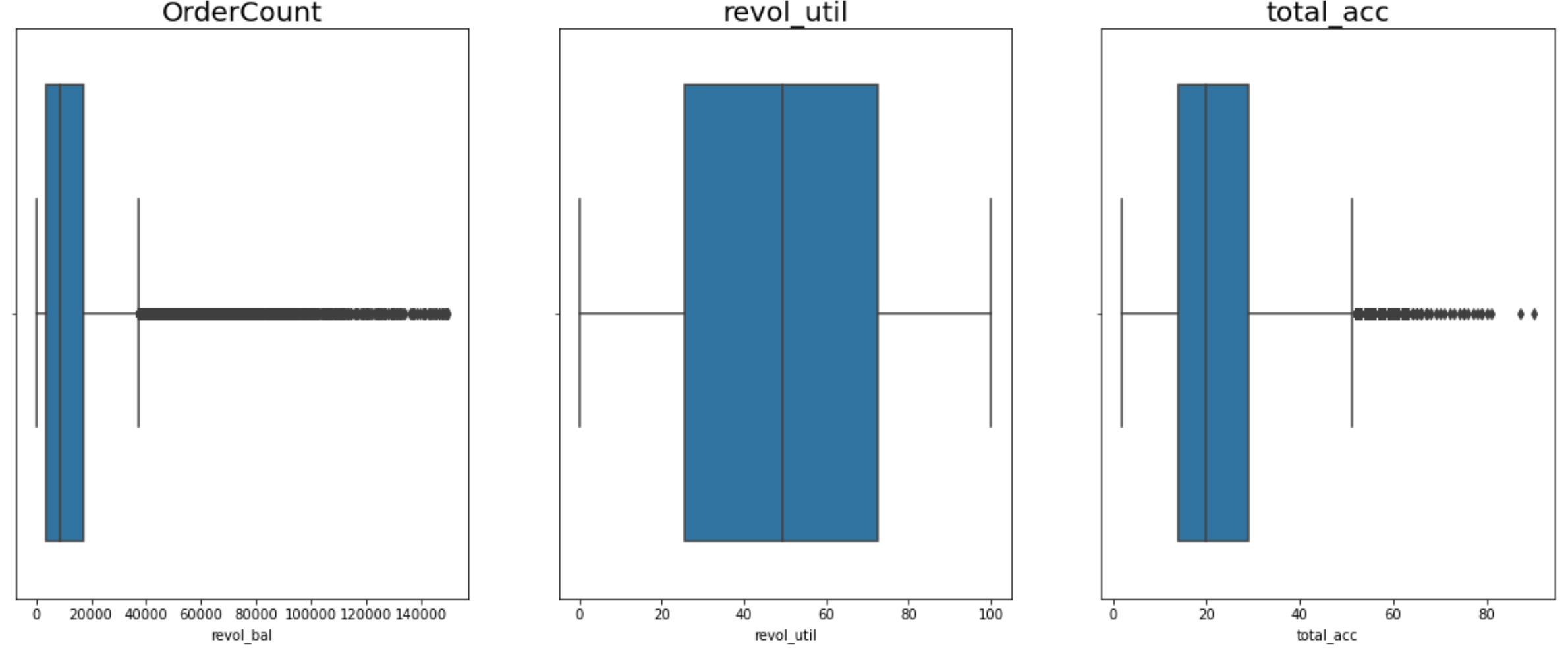
# 3.4. Handling Outliers:

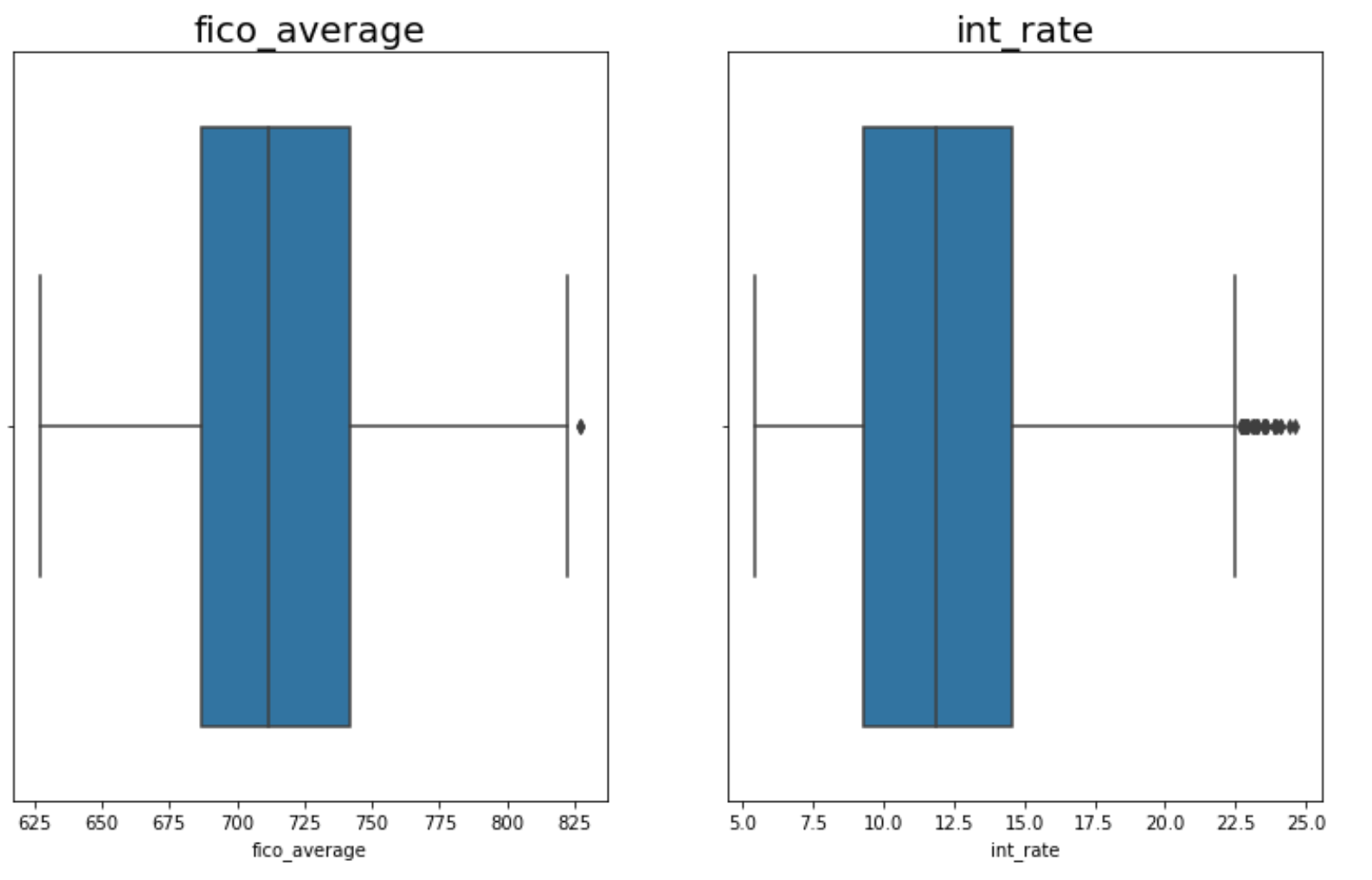
# 3.4.1 Detecting Outliers

We detected outliers by using box plot.









**List of features having outliers:**

loan\_amnt ,installment ,annual\_inc ,dti,inq\_last\_6mths ,open\_acc ,revol\_bal , revol\_util,total\_acc ,fico\_average, int\_rate

**Outlier Treatment:**

We applied capping technique to remove outliers for better performance of model.

We applied capping technique on installment , inq\_last\_6mnths,revol\_bal,open\_acc,total\_acc.

**3.5 Statistical Tests:**

Feature selection is done by performing **Chi square test** between target variable and other categorical variables and **ANOVA test** between target variable and continuous variables to check which features are important for our model building.

**3.6 Preparing data for modelling:**

**3.6.1 Encoding:**

We did encoding for converting categorical variables into dummy variables.

We have encoded values under column grade as follow:

**{'B'G':7,'F':6,'E':5,'D':4,'C':3,'B':2,'A':1}**

We have encoded values under column emp\_length as follow:

**< 1 year':1,'1 year':2,'2 years':3,'3 years':4, '4 years':5,'5 years':6,'6 years':7, '7 years':8,**

**'8 years':9,'9 years':10,'10+ years':11**

We have encoded values under loan status as follow:

**'Fully Paid':0,'Charged Off':1**

We have transformed “grade” and “emp\_length” and loan status to integer values by doing manual label encoding. This has been done for the ease of model training.

We used .get\_dummie() for data manipulation. It converted categorical variables into dummy variables and then we did scaling by using standard scaler on all the continuous variables to scale all the variables in the range of -1 to 1 and concatenated the encoded variables and scaled variables.

We saved the new analysed dataset into “modeldata.csv”.

**Chapter 4: MODELLING**

**4.1 Model preparation**

Before preparation of the model we have dropped the target variable and then went further for the modelling part.

**4.1.1 Decision Tree:**

Our base model is decision tree.

Decision Trees can be used as classifier or regression models.

A tree structure is constructed that breaks the dataset down into smaller subsets eventually resulting in a prediction. There are decision nodes that partition the data and leaf nodes that give the prediction that can be followed by traversing simple IF..AND..AND….THEN logic downs the nodes.

The root node (the first decision node) partitions the data based on the most influential feature partitioning. There are 2 measures for this, Gini Impurity and Entropy.

By using decision tree classifier we got the feature information of every variable in the dataset.

By this we filtered out the only important features from the dataset and rest of the columns we dropped.

**Checking for multicollinearity using VIF:**

One way to measure multicollinearity is the variance inflation factor (VIF), which assesses how much the variance of an estimated regression coefficient increases if your predictors are correlated. If no factors are correlated, the VIFs will all be 1.

By using VIF we checked for multicollinearity in the dataset and we found that grade, loan\_amount, installment has been having multicollinearity in the dataset so we dropped those feature.

**Splitting the Dataset:**

The scaled data is split into train and test data. The Train data set will be used to build the model and the test data will be used to test the performance of the trained model

The train data contained 70% data and the test data is 30%.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3 , random\_state=10)

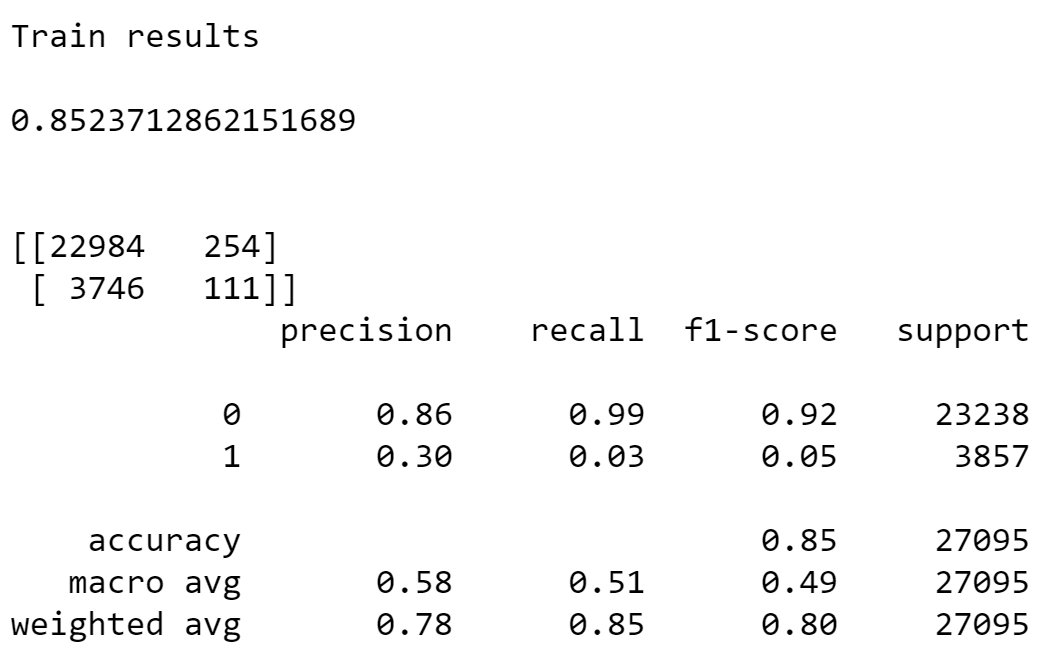
**4.1.2 Logistic Regression:**

**Fitting model to the data:**

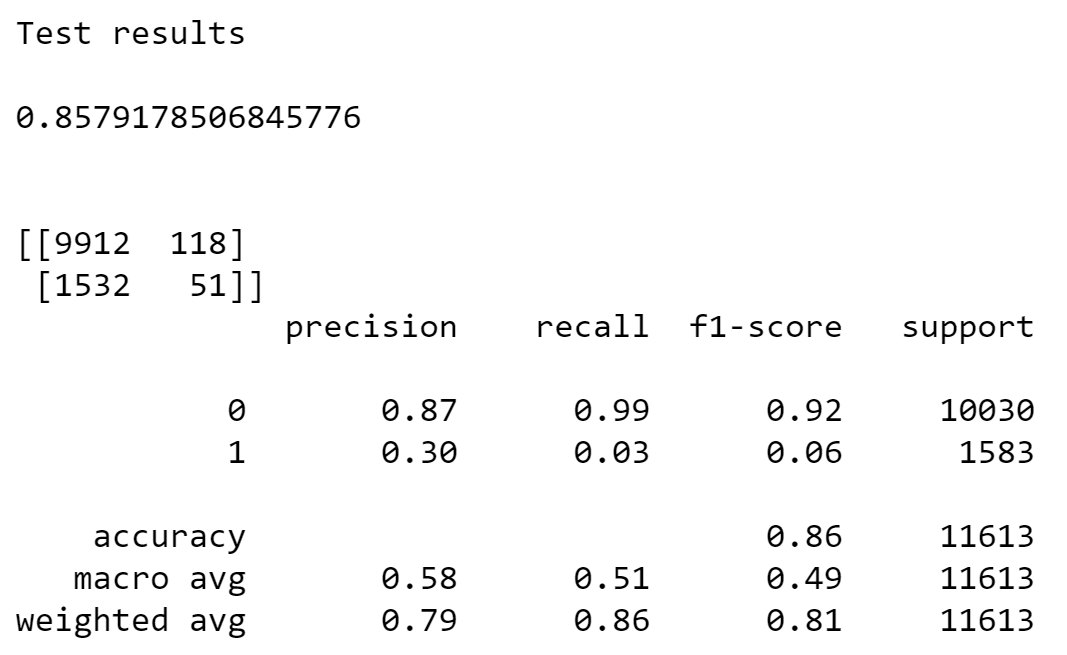
Logistic regression is a fundamental classification technique. It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression.

Logistic regression is fast and relatively uncomplicated, and it’s convenient for you to interpret the results. Although it’s essentially a method for binary classification, it can also be applied to multiclass problems.

Performance of logistic regression model on train data gave the following result:



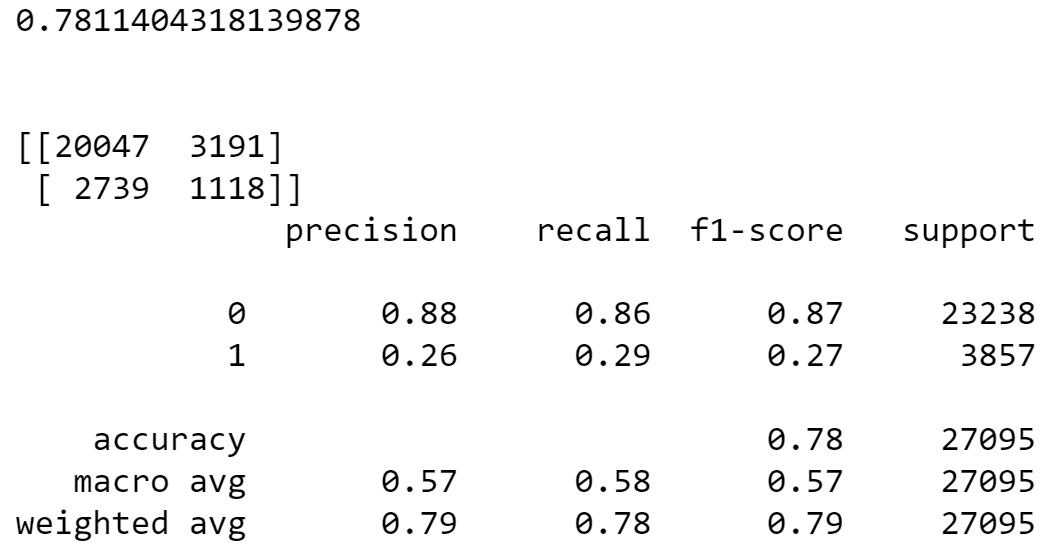
Performance of Logistic Regression model on test data gave the following result:



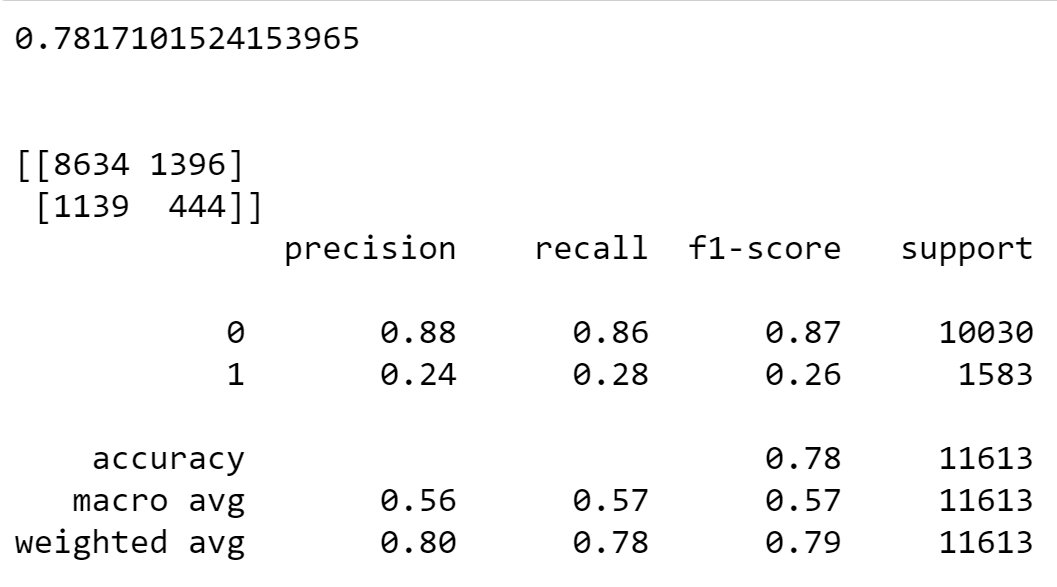
**4.1.3 Naive Bayes:**

The Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes’ theorem with the “naive” assumption of conditional independence between every pair of features given the value of the class variable.

Performance of Naïve Bayes model on train data gave the following result:



Performance of Naïve Bayes model on test data gave the following result:



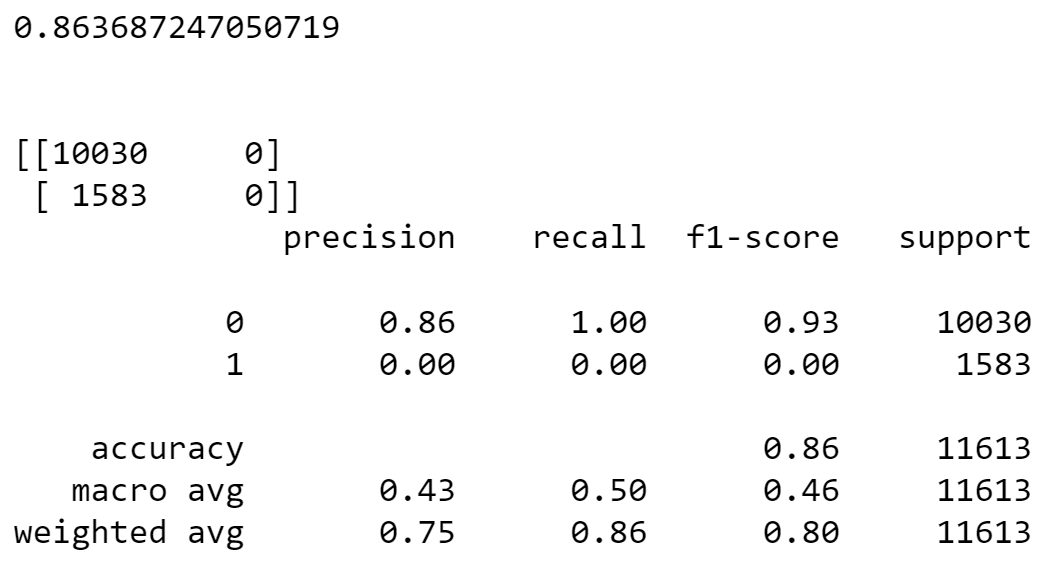
**4.1.4 SVM Model**

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers’ detection. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

Performance of SVM model on train data gave the following result:



Performance of SVM model on test data gave the following result:



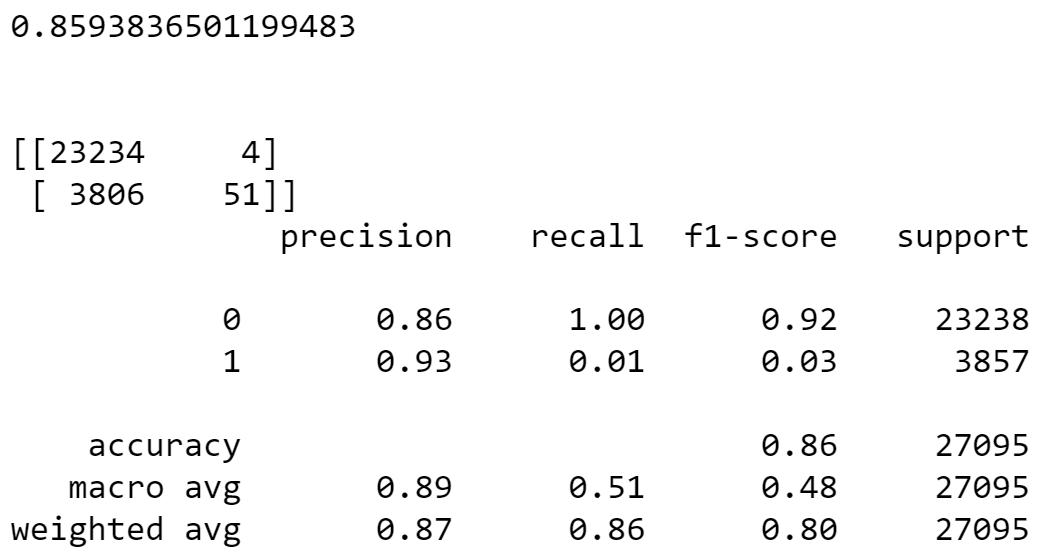
## 4.1.5 XGBoost Model

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

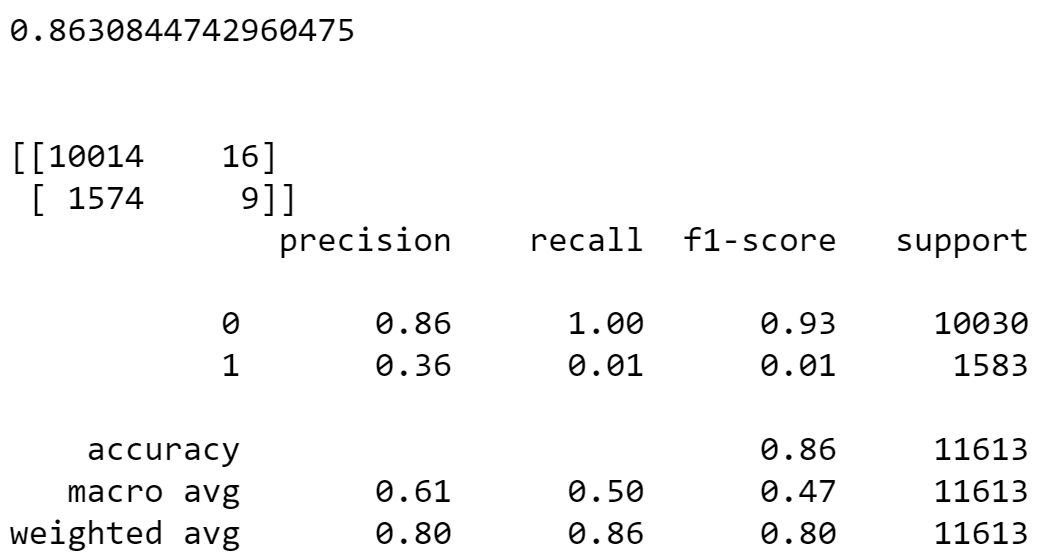
The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e. how far the model results are from the real values. The most common loss functions in XGBoost for regression problems is reg:linear, and that for binary classification is reg:logistics.

Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction, and XGBoost is one of the ensemble learning methods. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancels out and better one sums up to form final good predictions.

Performance of xgboost model on train data gave the following result:



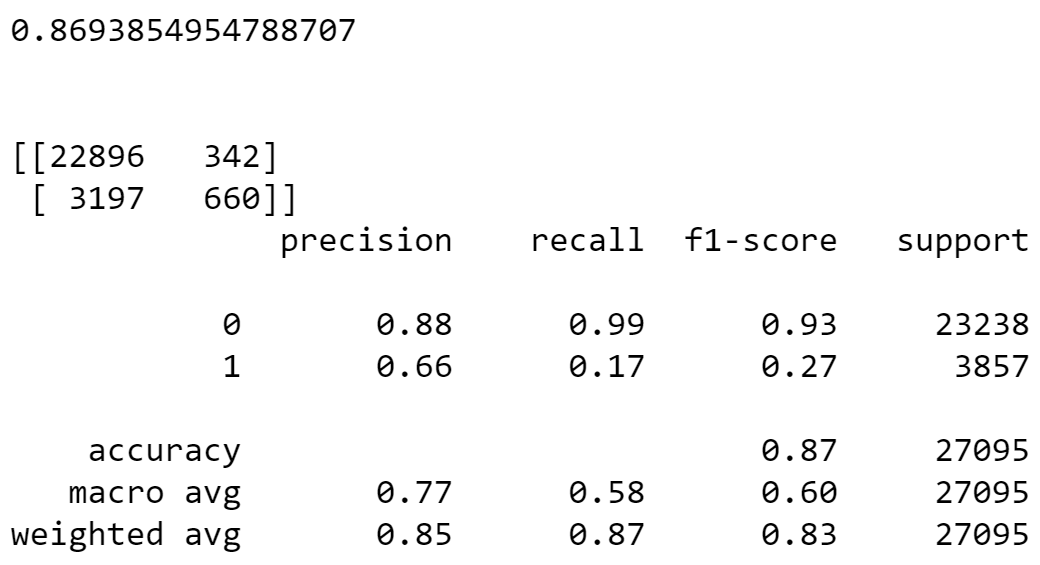
Performance of xgboost model on test data gave the following result:



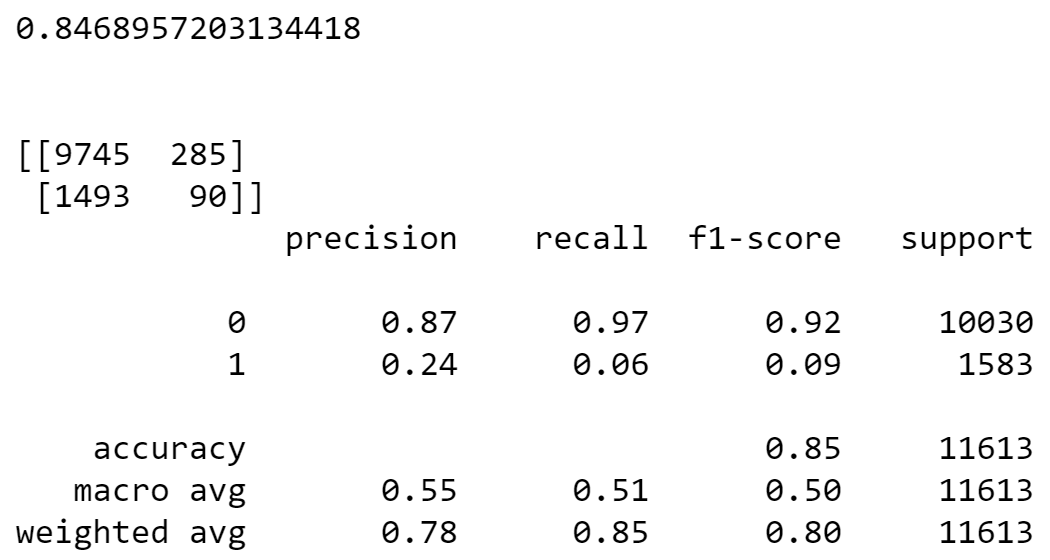
## 4.1.6 KNN Classifier

The K-nearest neighbours (KNN) algorithm is a type of supervised machine learning algorithms. [KNN](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm) is extremely easy to implement in its most basic form, and yet performs quite complex classification tasks. It is a lazy learning algorithm since it doesn't have a specialized training phase. Rather, it uses all of the data for training while classifying a new data point or instance. KNN is a non-parametric learning algorithm, which means that it doesn't assume anything about the underlying data. This is an extremely useful feature since most of the real world data doesn't really follow any theoretical assumption.

Performance of KNN classifier model on train data gave the following result:



Performance of KNN classifier model on test data gave the following result:

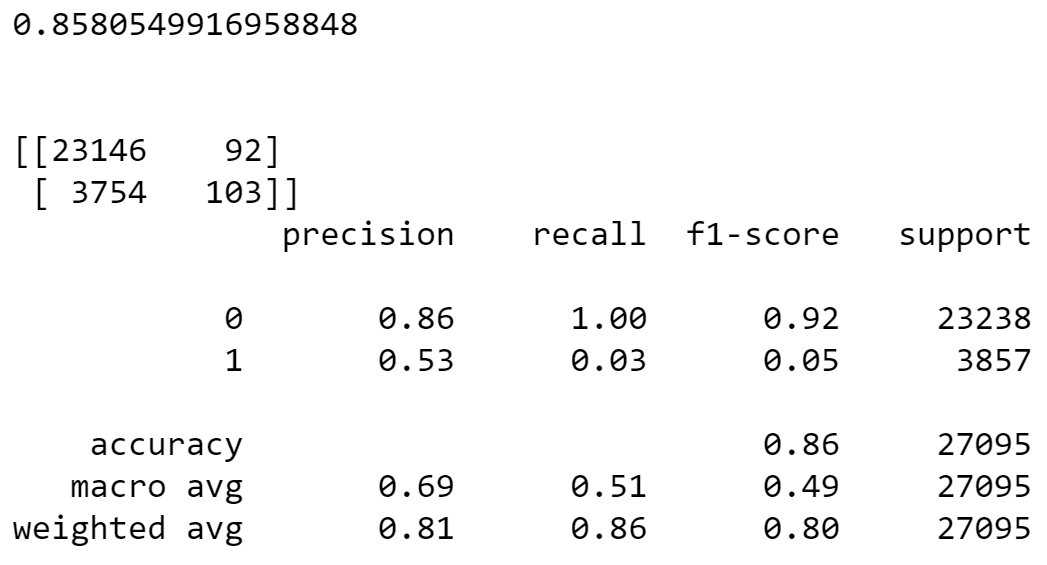


## 4.1.7 AdaBoost Classifier

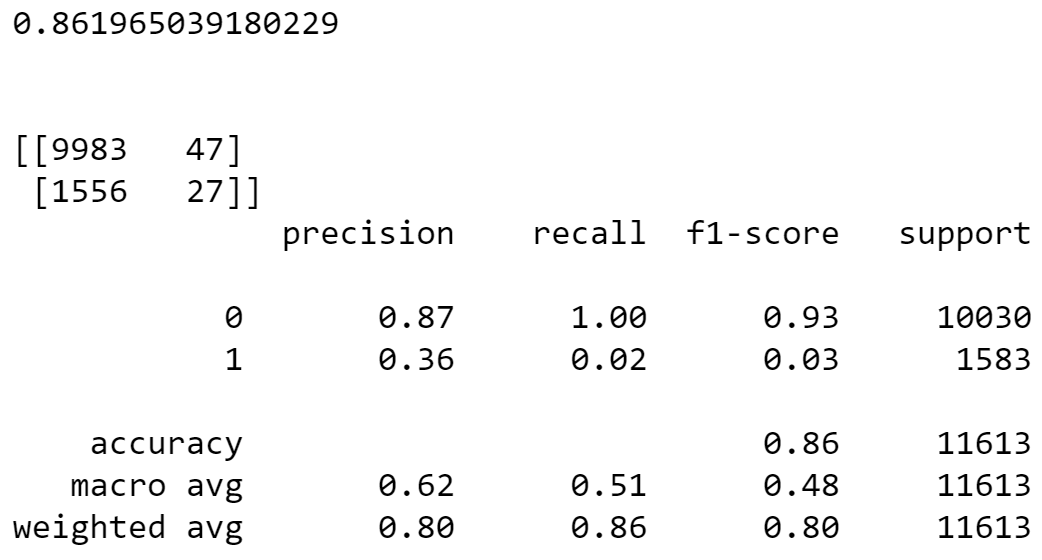
AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. Adaboost should meet two conditions:

1. The classifier should be trained interactively on various weighed training examples.
2. In each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

Performance of Adabooster classifier model on train data gave the following result:



Performance of Adabooster classifier model on test data gave the following result:



**4.1.8 Random Forest**

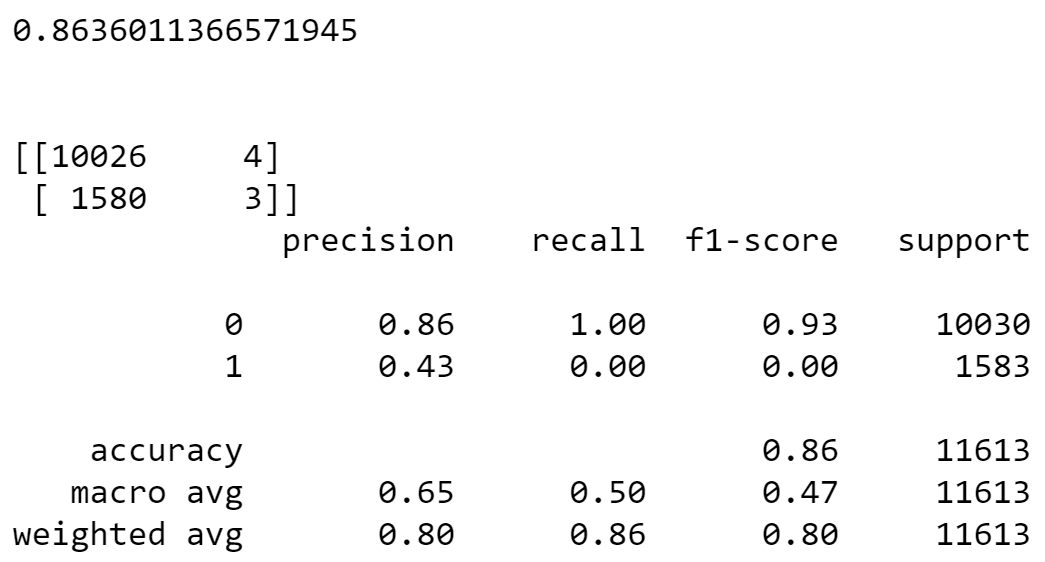
Random forest is a [supervised learning algorithm](https://builtin.com/data-science/supervised-learning-python). The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

**Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.**

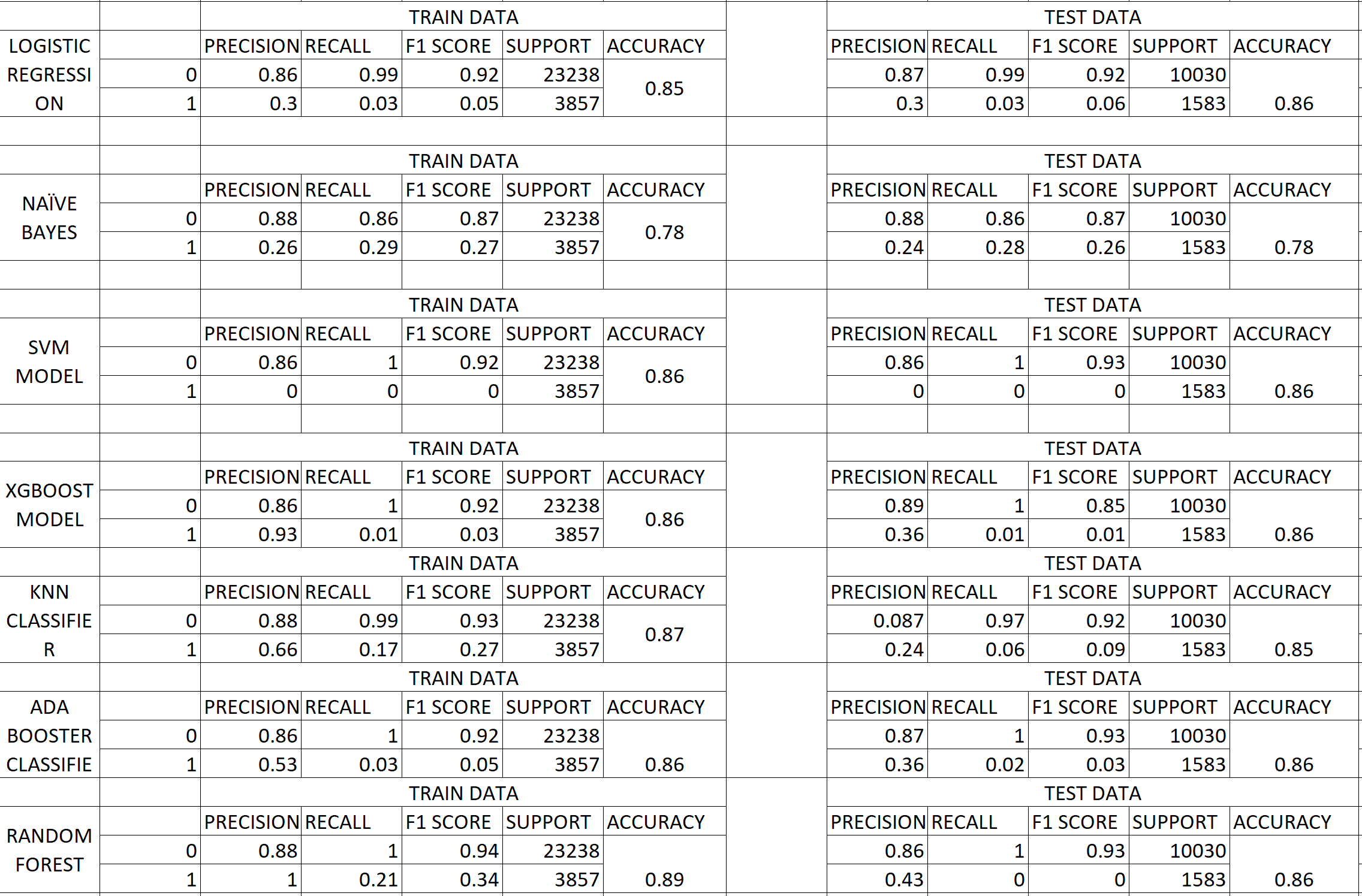
Performance of Random forest model on train data gave the following result:



Performance of Random forest model on test data gave the following result:



**Comparison and Conclusion:**

****

### From the above results we are taking Random Forest and AdaBoost for further tuning.

**4.1.9 SMOTE:**

SMOTE stands for Synthetic Minority Oversampling Technique. This is a statistical technique for increasing the number of cases in your dataset in a balanced way. SMOTE takes the entire dataset as an input, but it increases the percentage of only the minority cases.

We applied smote because of imbalance in the target.

**4.2 Grid Search**

Grid-search is used to find the optimal hyperparameters of a model which results in the most ‘accurate’ predictions.

from sklearn.model\_selection import GridSearchCV

**4.2.1 AdaBoost Model Tuning with SMOTE**

Tuning is the process of maximizing a model's performance without overfitting or creating too high of a variance. In machine learning, this is accomplished by selecting appropriate “hyperparameters.”

**Probability Tuning-**

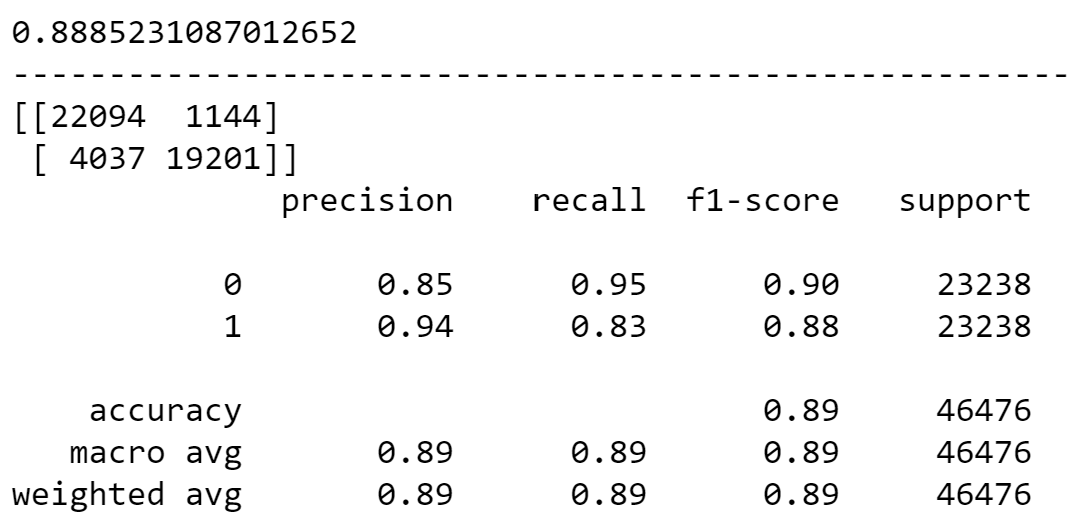
Probability tuning is the process to improve the recall value by changing the default value.

### By using grid search we got best parameters:

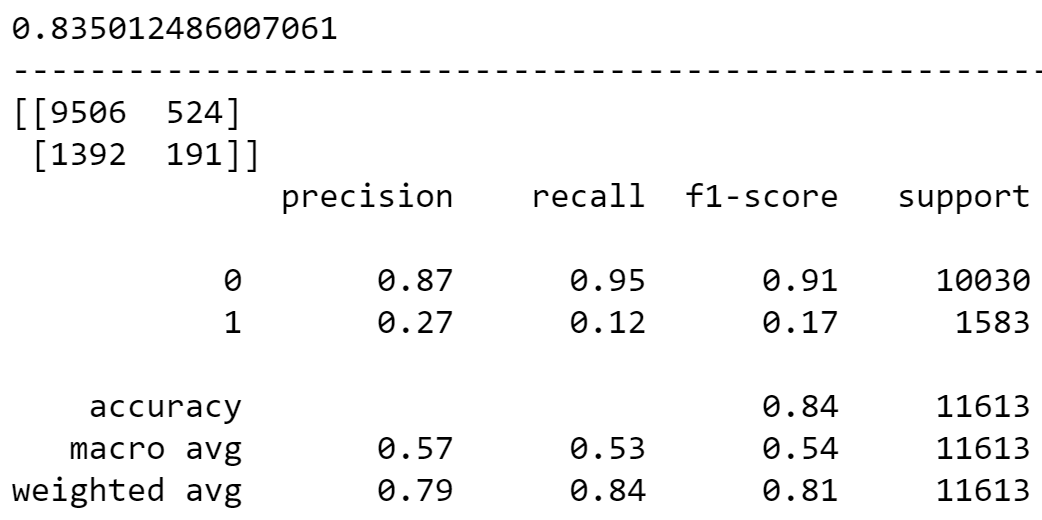
### best parameters - {'learning\_rate': 1.65, 'n\_estimators': 300}[¶](http://localhost:8888/notebooks/Downloads/modeling%20(3).ipynb#best-parameters---{'learning_rate':-1.65,-'n_estimators':-300})

### We built the new tuned AdaBoost model

Performance of New tuned AdaBoost model on train data gave the following result:

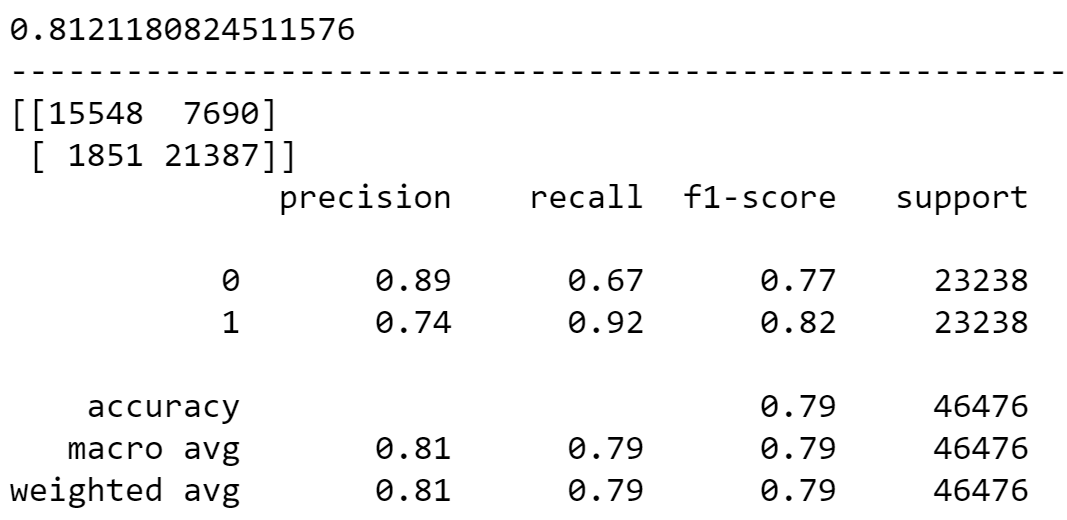


Performance of New tuned AdaBoost model on test data gave the following result:

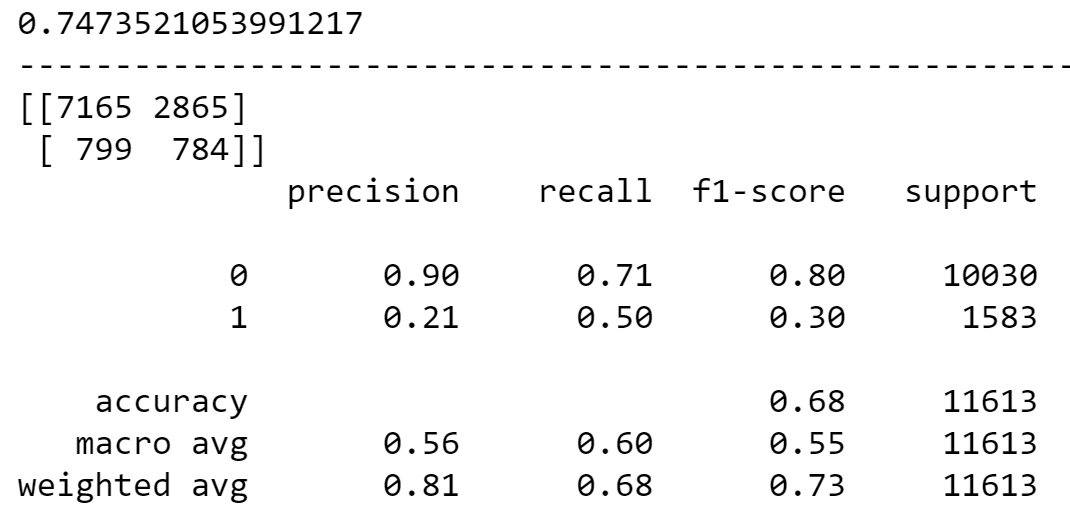


**4.2.2 AdaBoost Probability Tuning**

Post probability tuning train scores of AdaBoost tuned model.



Post probability tuning test scores of AdaBoost tuned model.



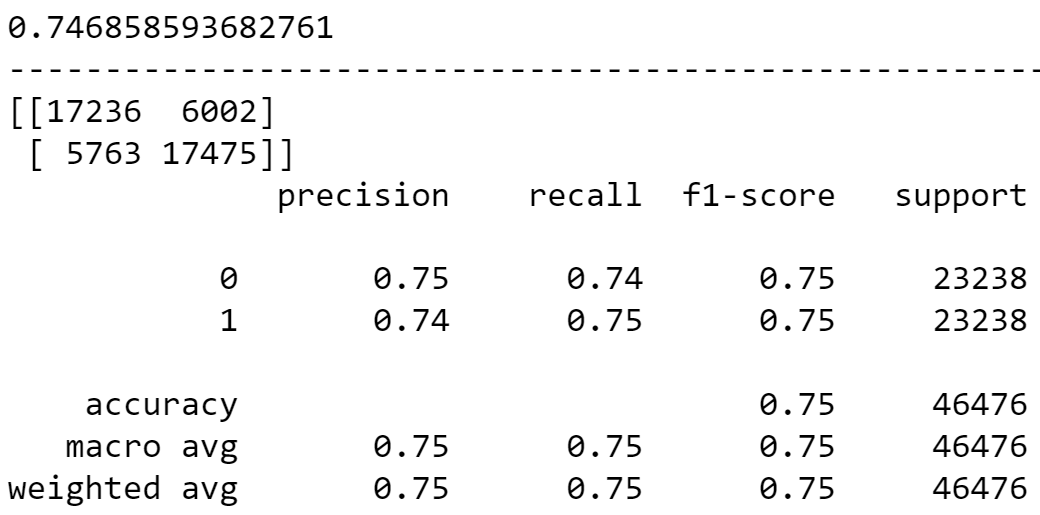
# 4.2.3 Random Forest Model Tuning with SMOTE

### By using grid search we got best parameters:

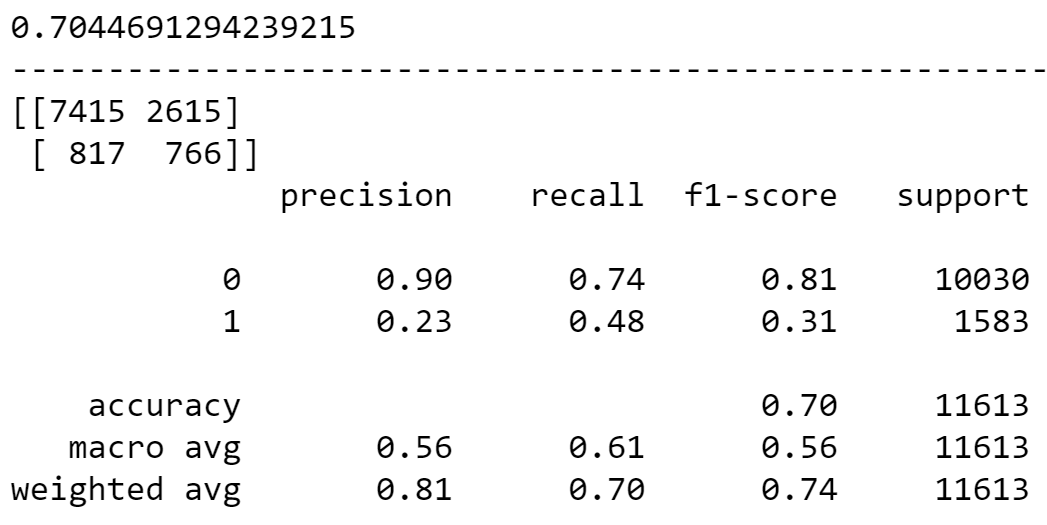
best parameters { 'bootstrap': True, 'max\_leaf\_nodes':40, 'n\_estimators':100 }

### We built the new tuned Random forest model

Performance of New tuned Random forest model on train data gave the following result:

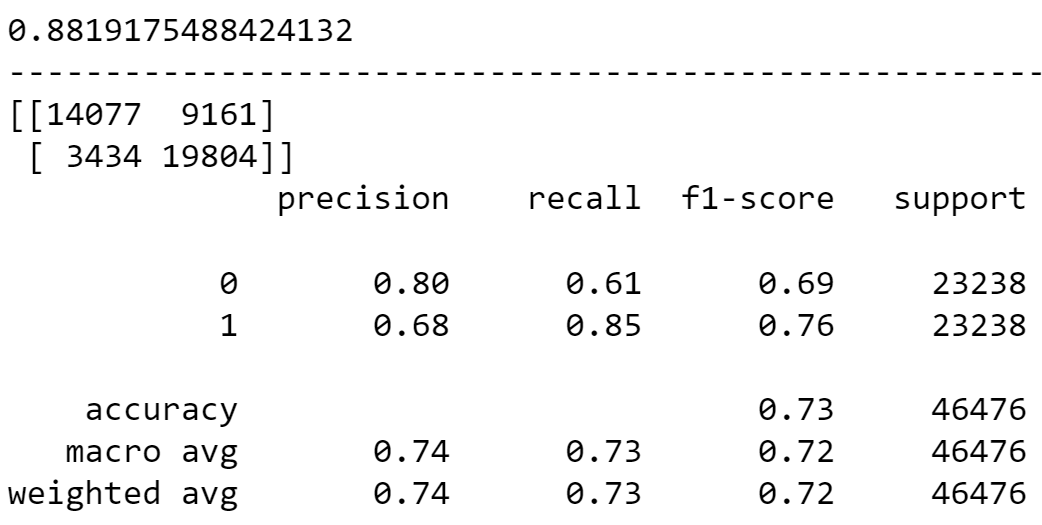


Performance of New tuned Random forest model on test data gave the following result:

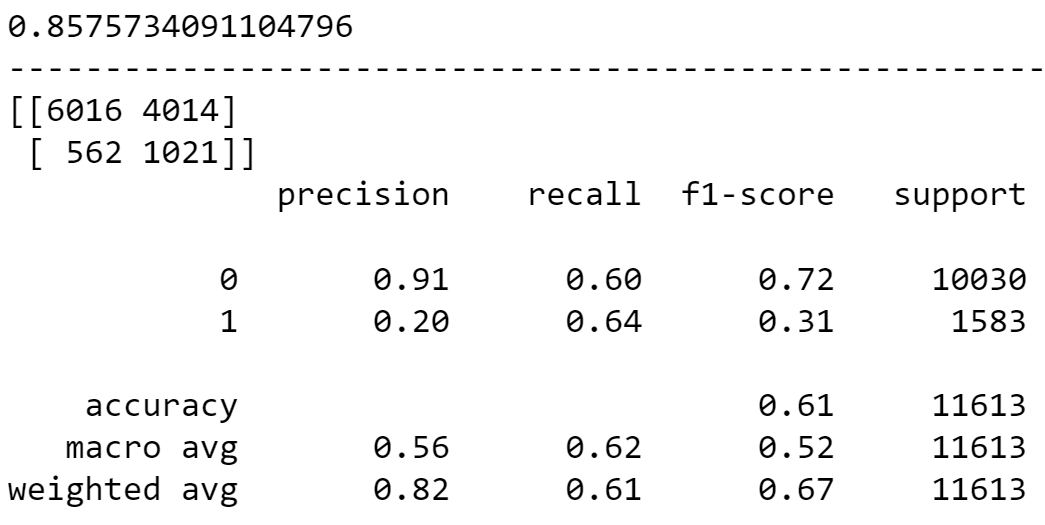


**4.2.4 Random Forest Probability Tuning**

Post probability tuning train scores of Random forest model.



Post probability tuning test scores of Random forest model.



**4.2.5 Comparison and conclusion:**

****

**Note**: Random forest with SMOTE gives the best test and train result for both recall and accuracy.

**4.3 Cross** **Validation**

The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset.

Cross score validation of tuned random forest model on train data

**[0.91006885, 0.91415663, 0.90942341, 0.90877797, 0.88403614,**

**0.86359725, 0.88035292, 0.8779858 , 0.88056811, 0.91822681]**

The average train accuracy of the model is: **0.8947193875566825**

Cross score validation of tuned random forest model on test data

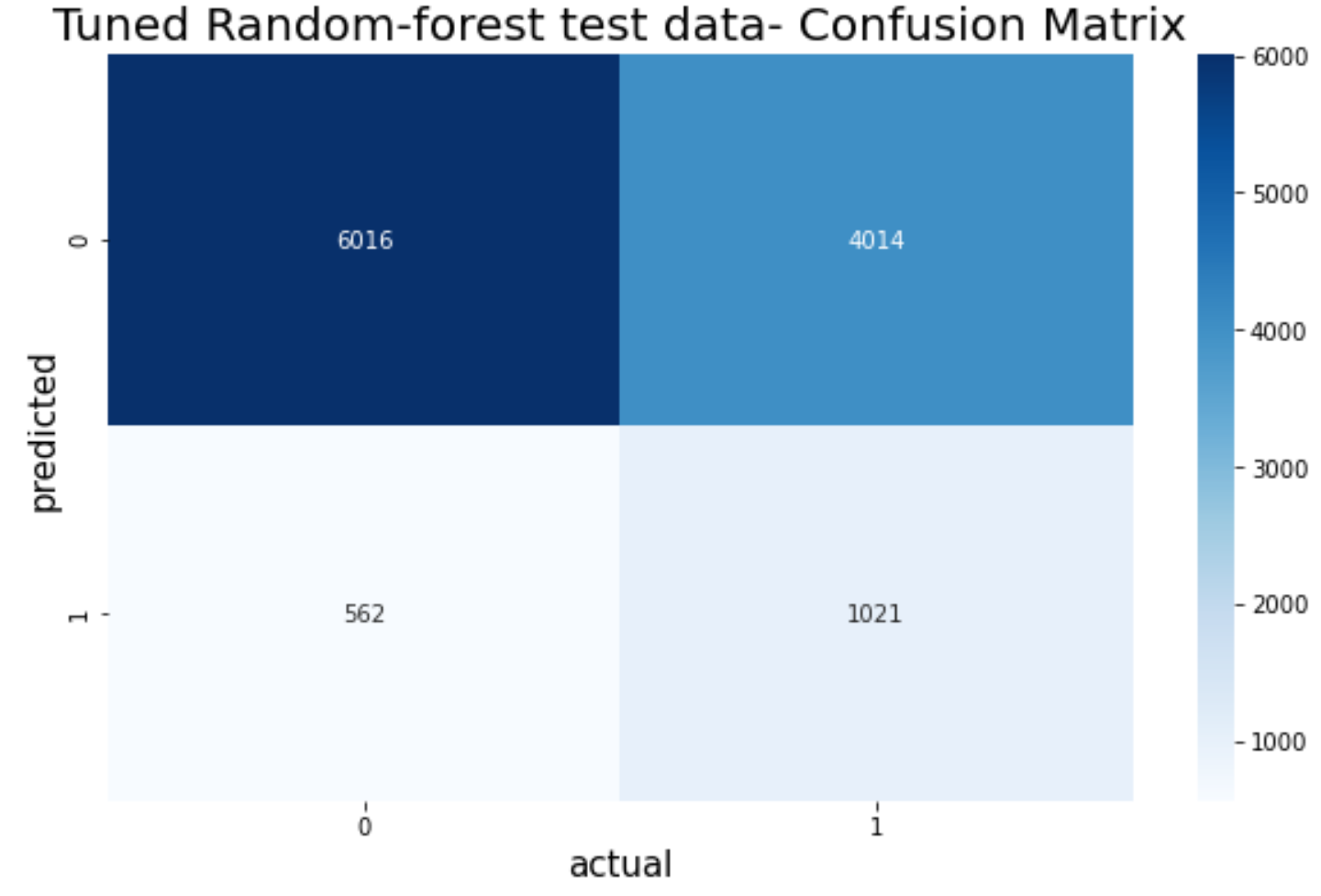
**[0.91135972, 0.90533563, 0.91049914, 0.90353144, 0.87855297,**

**0.8828596 , 0.89061154, 0.90611542, 0.90439276, 0.91559001]**

The average test accuracy of the model is: **0.9008848239024759**

**4.4 Model Prediction:**

Confusion Matrix of tuned random forest test data:



**Key points to consider for loan approval:**

We suggest the management to use tuned random forest model to predict loan defaulters.

* Borrowers with high interest rate are more likely to default.
* Borrowers with higher annual income are less likely to default.
* Borrowers who have higher revolving balance amount left are likely to default.
* Borrowers with high debt to income ratio are more likely to default.