MICRO – CREDIT DEFAULTER PROJECT

Machine learning model

By

SOHAM BANDGAR

Problem statement

Problem Statement:

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.

Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

The sample data is provided to us from our client database. It is hereby given to you for this exercise. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

Understanding of problem statement

A Micro Finance Institute (MFI) is an financial organization who focus and target the unbaked low income population, living in remote areas who does not have much source of income. They have global outreach of 200 million clients with $70 billion as an outstanding loans that is why these micro finance institutes are widely accepted as poverty reduction tools.

These MFIs have understood the importance of communication in a person’s daily life & how it affects the life of person hence to improve the basic need of communication of financially poor low income families living in remote areas they come up with a concept of providing mobile balance credits to the people, collaborating with a Telecom industry who provides wireless telecommunication network & are good with providing better offer at low price.

Now there is a challenge of default case where a user don’t pay back the loaned mobile balance back. To encounter this problem organization wants a machine learning model that can predict the default case so that they can prevent their financial loss & can improve their services.

Understanding the data

- The micro credit defaulter data is of a telecom industry, describing the recharges of customer mobile balance, average balance of 30 day & 90 days, average spent of balance in 30 days & 90 days, loan opted by customer in 30 days & 90 days, circle of telecom, mobile number etc.
- The data consist of 36 such columns and had record of over 2 lack customers.
- Data has no missing values.
- This data has originated from the client data base.
- The format of data was in csv format.
- The data had 3 categorical column & out 2 columns were object data type.
- Rest all of the columns were continuous data having integer or float data type.

Software, tools & Libraries used in project

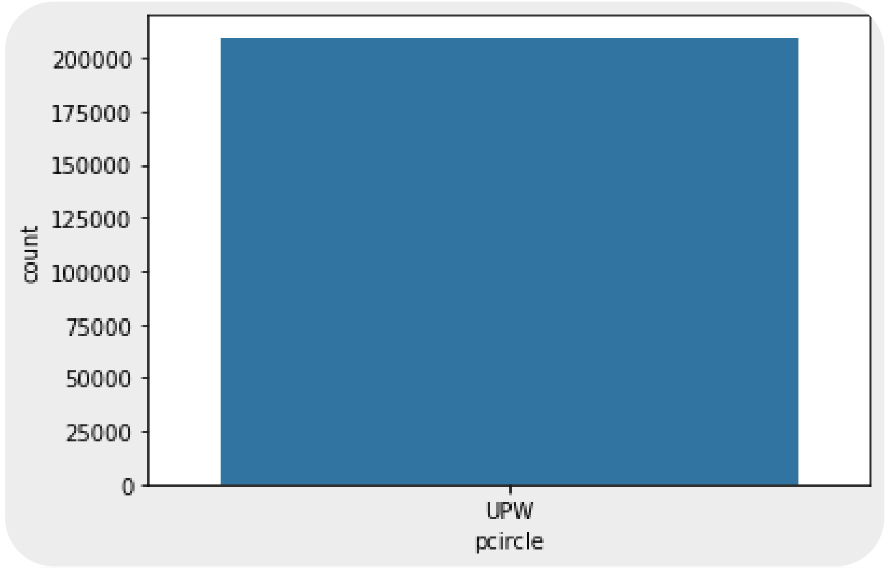
- Python Programming language in Jupyter NoteBook is used for this project.
- Anaconda software provided all of the above mentioned thing in one platform.
- Pandas Library is used to read the data in python programming language.
- Numpy Library is used for mathematical implementation in the dataset.
- Matplotlib & Seaborn libraries used for plotting and visualizing the statistical analysis of the data
- Scikit Learn is used for model building, model evaluation & Hyper parameter tuning of the model.
- Pickle used to save the model in local system.

Exploratory data analysis
&
Understanding of visualization

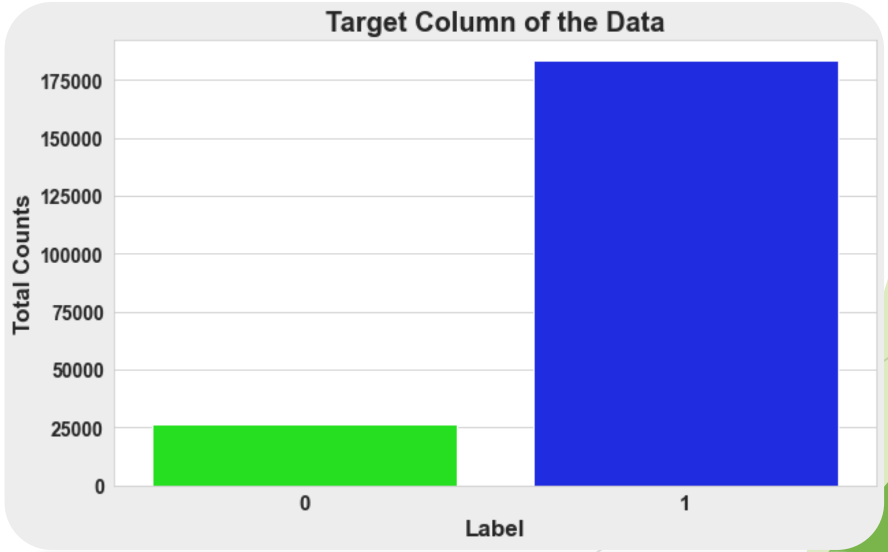
- For the exploratory data analysis and visualization of the data we used matplotlib library's pyplot package & seaborn library.
- For the understanding of categorical data we used count plot.
- For the continuous data we used distribution plot.
- We used box plot to see & understand the outliers in the data set.
- For the correlation of the data we used heat map of seaborn.
- We also used scikit's roc plot to visualize the area under the curve to evaluate the machine learning models.
- Based on these steps of eda & visualization we took some assumption & processed necessary steps to complete this project which we are going to discuss in this power point presentation.

EDA & visualization of categorical data & conclusion.

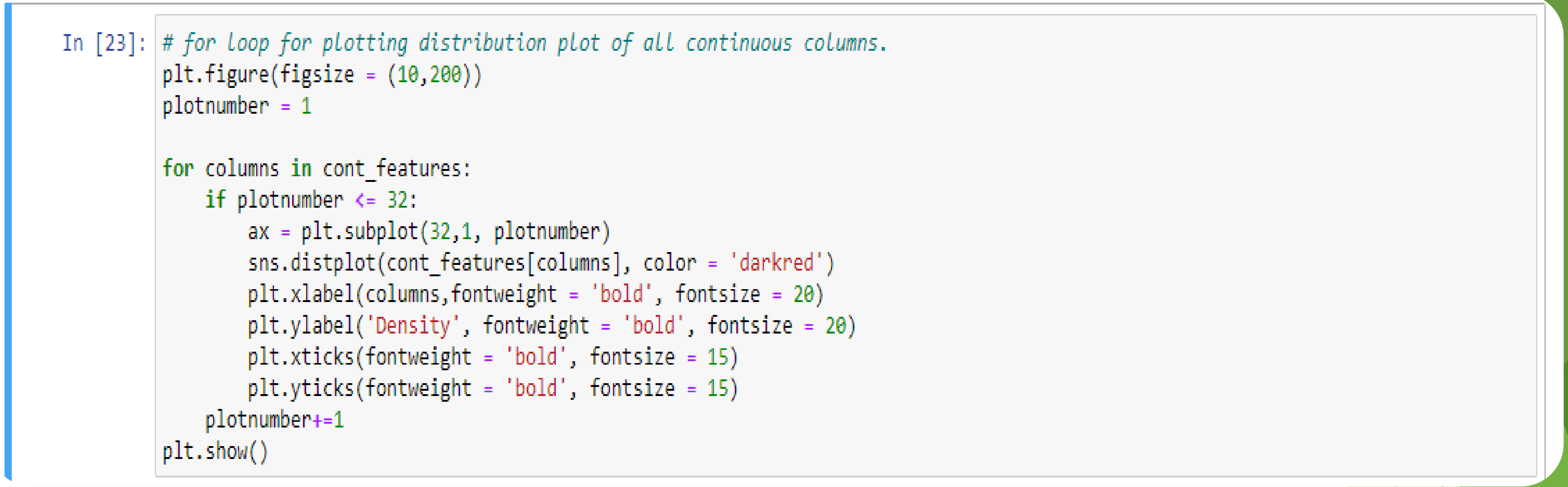
- We found in pcircle column only 1 category is present named **upw.**
- From this we concluded that this column is neutral for each row & will play no role in defaulter prediction.
- Hence we decided to drop this column.



- From the data description of micro credit data we got to know that Label column is our class or target column representing default and not default cases.
- Where 0 was or default case & 1 was for not default case.
- From the data visualization we analyzed that class data is imbalanced data.
- Hence we concluded to balance the data by over sampling of class data

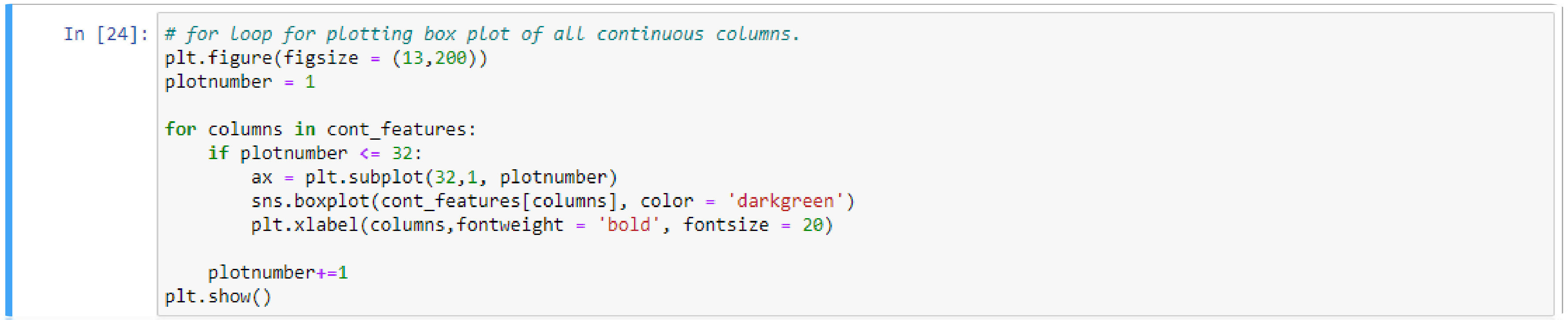


Code for visualization distribution plot of all the continuous column at once



```
In [23]: # for loop for plotting distribution plot of all continuous columns.
         plt.figure(figsize = (10,200))
         plotnumber = 1

         for columns in cont_features:
             if plotnumber <= 32:
                 ax = plt.subplot(32,1, plotnumber)
                 sns.distplot(cont_features[columns], color = 'darkred')
                 plt.xlabel(columns,fontweight = 'bold', fontsize = 20)
                 plt.ylabel('Density', fontweight = 'bold', fontsize = 20)
                 plt.xticks(fontweight = 'bold', fontsize = 15)
                 plt.yticks(fontweight = 'bold', fontsize = 15)
             plotnumber+=1
         plt.show()
```

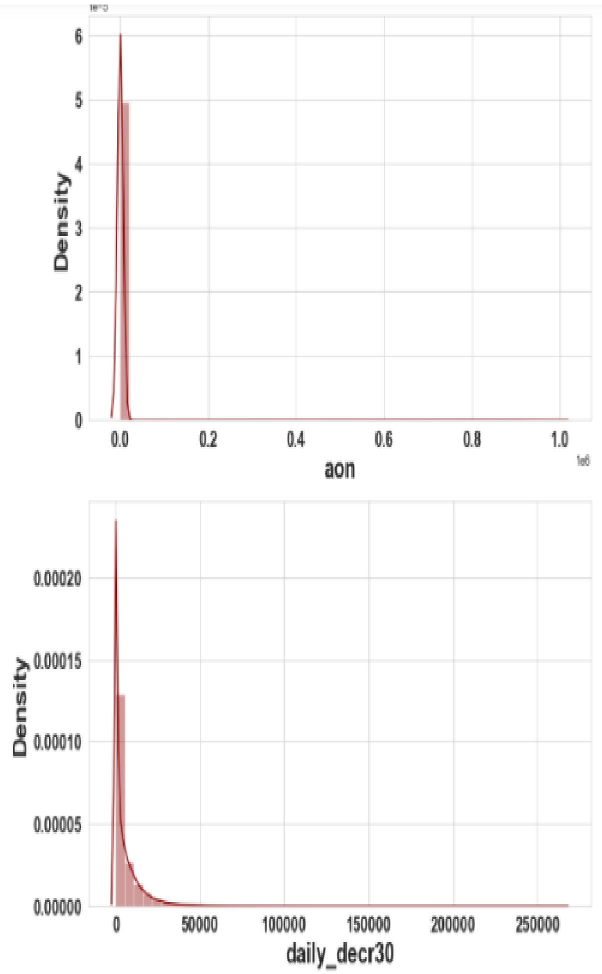
Code for visualization box plot of all the continuous column at once



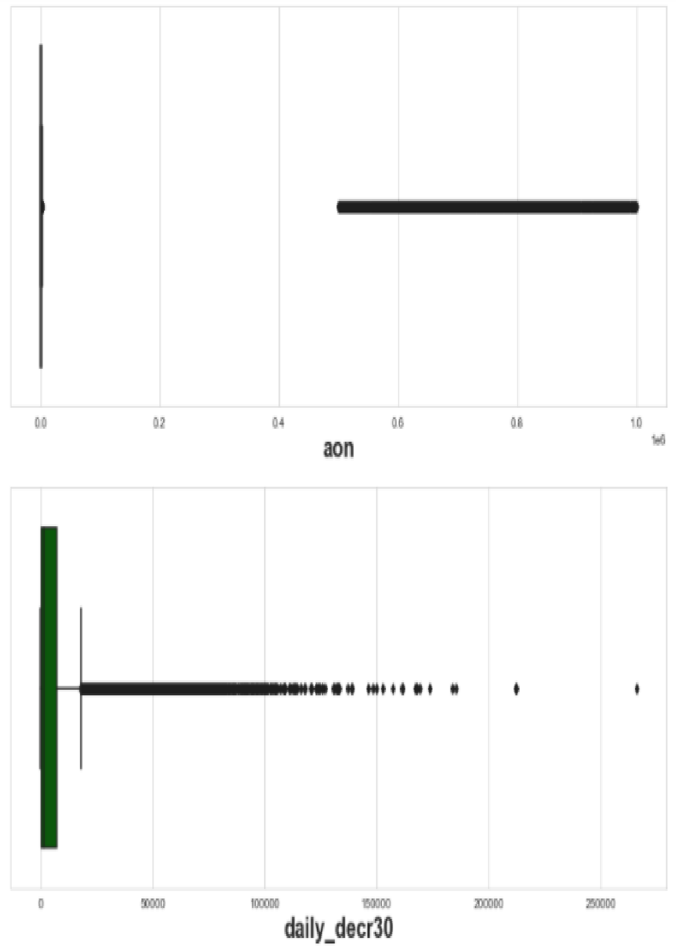
```
In [24]: # for loop for plotting box plot of all continuous columns.
         plt.figure(figsize = (13,200))
         plotnumber = 1

         for columns in cont_features:
             if plotnumber <= 32:
                 ax = plt.subplot(32,1, plotnumber)
                 sns.boxplot(cont_features[columns], color = 'darkgreen')
                 plt.xlabel(columns,fontweight = 'bold', fontsize = 20)

             plotnumber+=1
         plt.show()
```

Visualization of count plot & box plot

Distribution plot for “aon” & “daily_dcr30”.

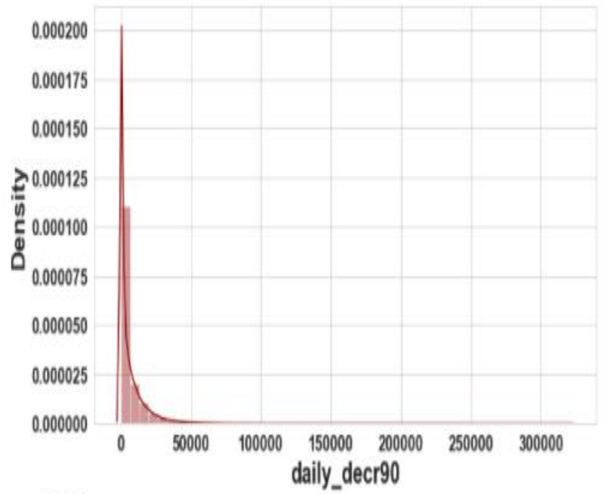


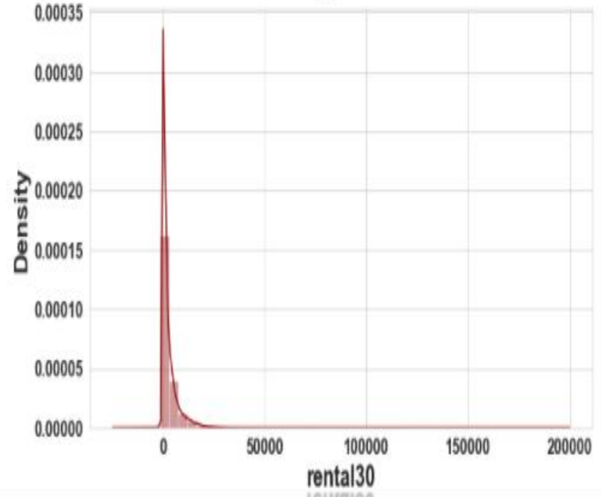
box plot for “aon” & “daily_dcr30”.



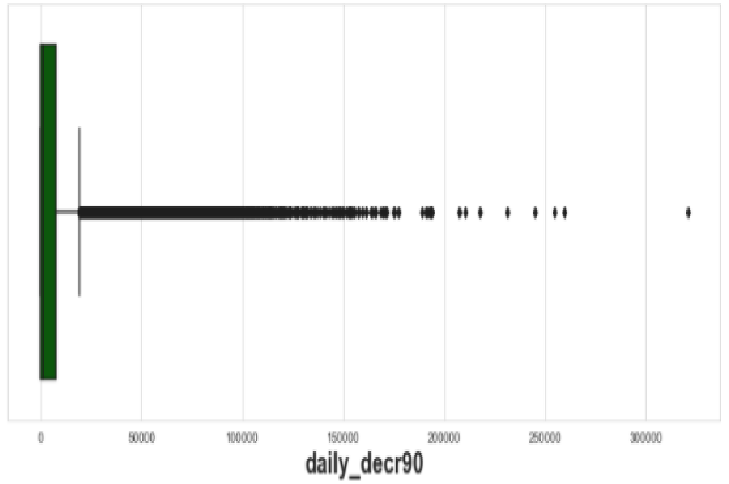
Visualization of count plot & box plot

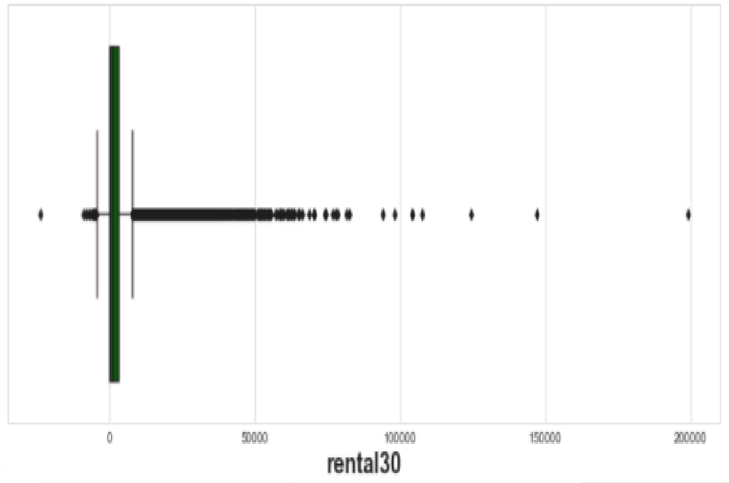
Distribution plot for "daily_dcr90" & "rental30".





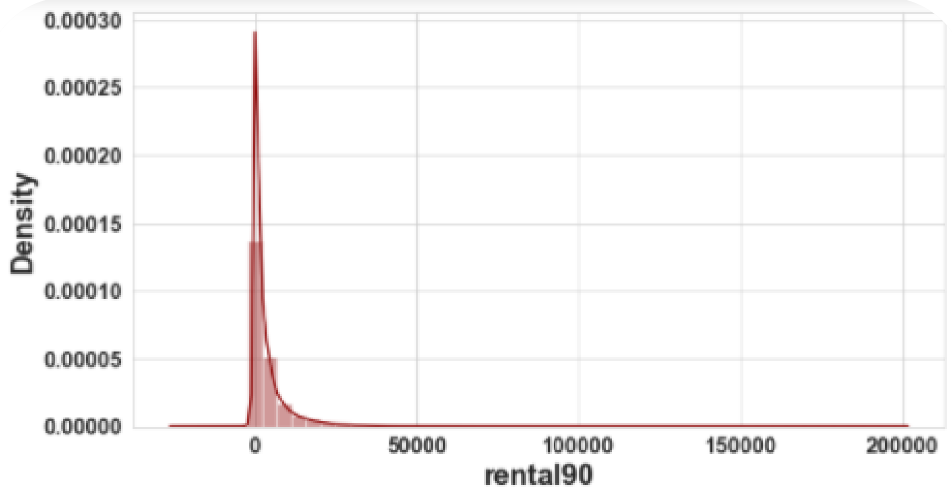
box plot for "daily_dcr90" & "rental30".

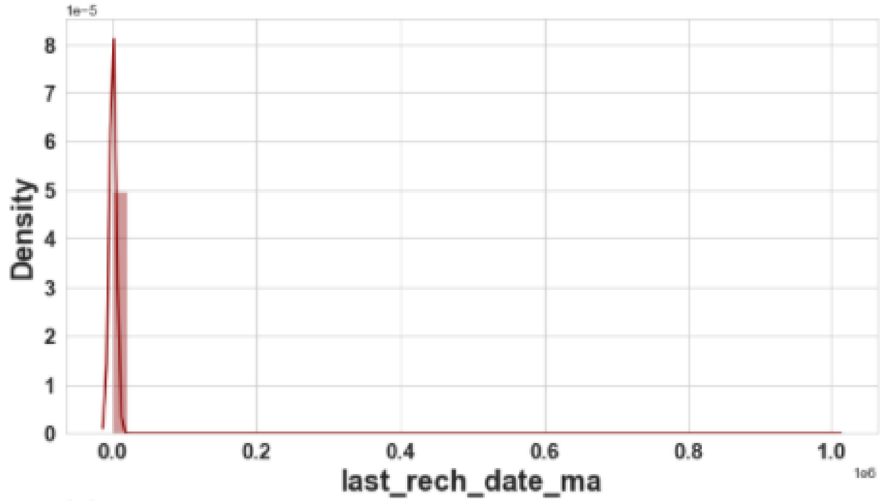




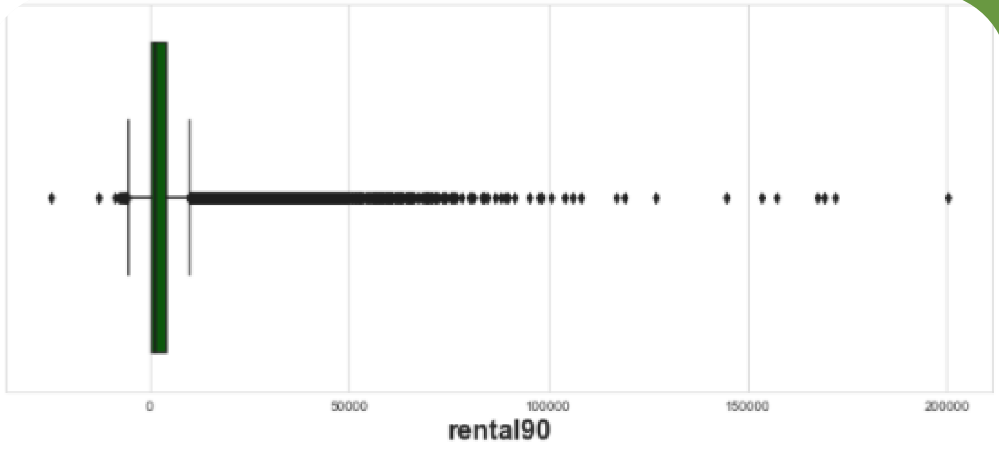
Visualization of count plot & box plot

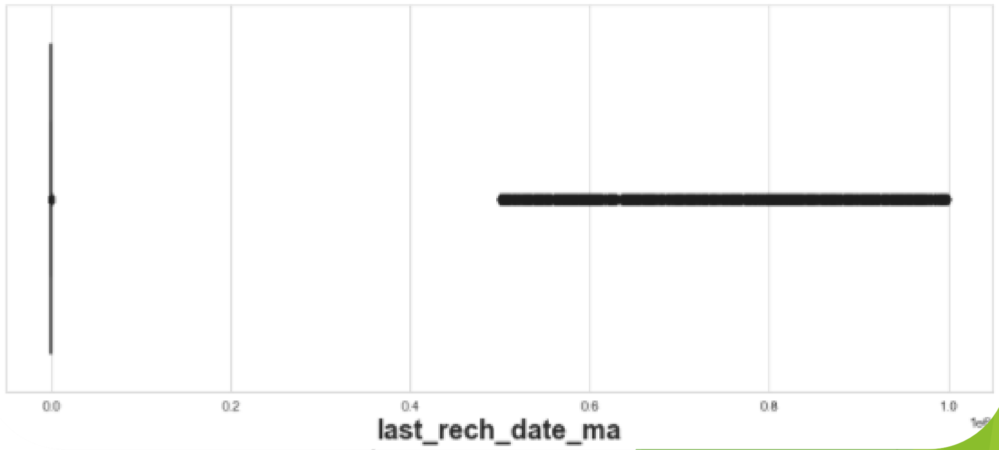
Distribution plot for “rental90” & “last_rech_date_ma”.





box plot for “rental90” & “last_rech_date_ma”.

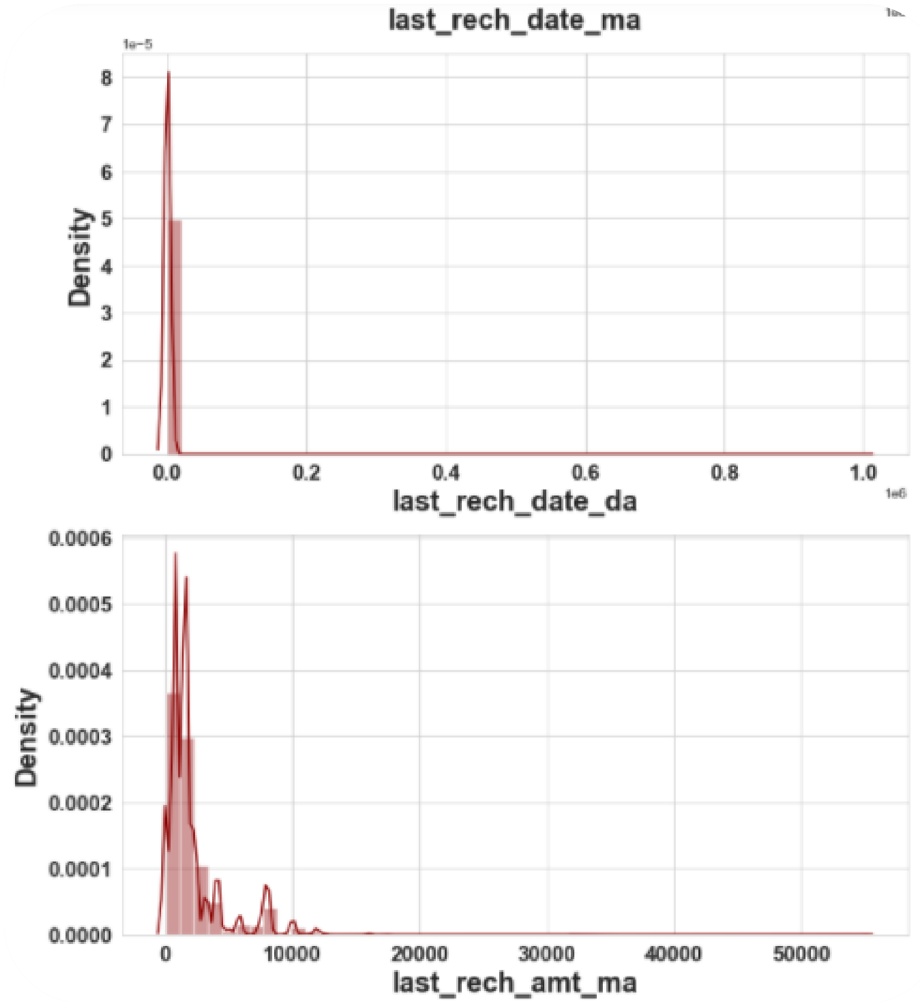


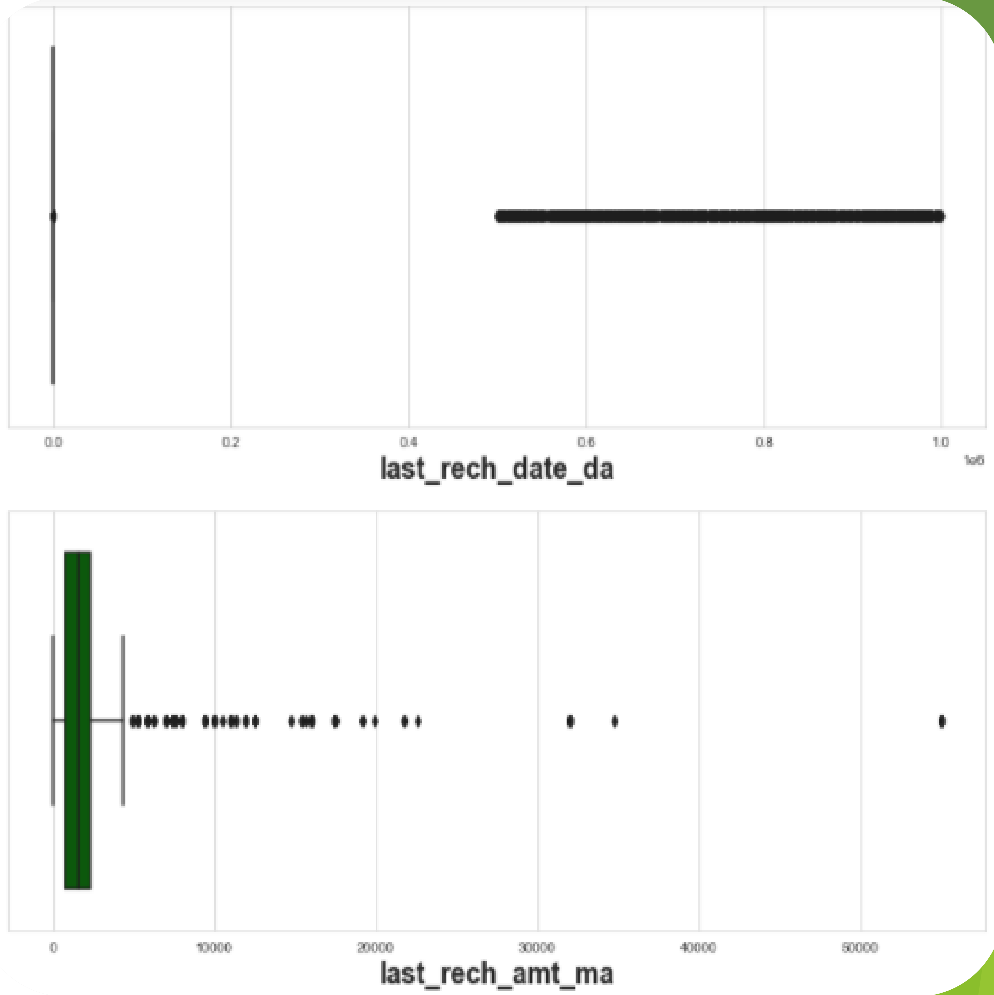


Visualization of count plot & box plot

Distribution plot for “rental90” & “last_rech_date_ma”.

box plot for “rental90” & “last_rech_date_ma”.

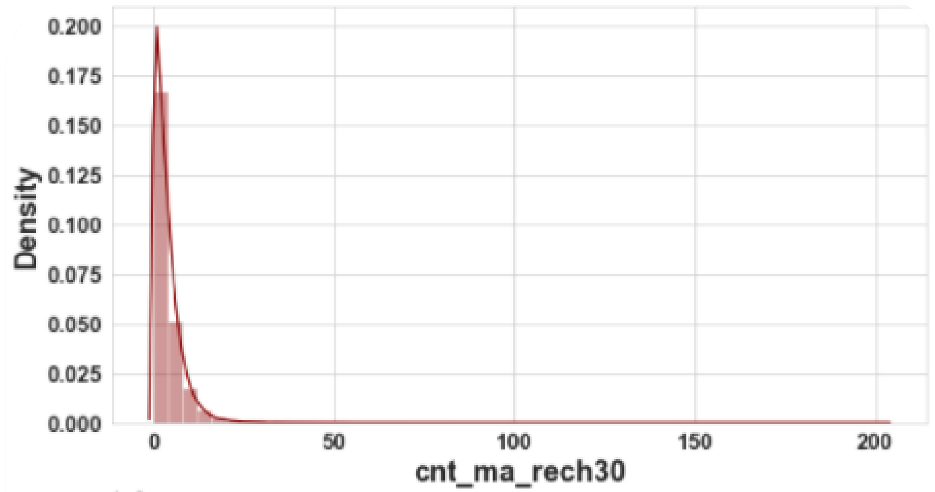


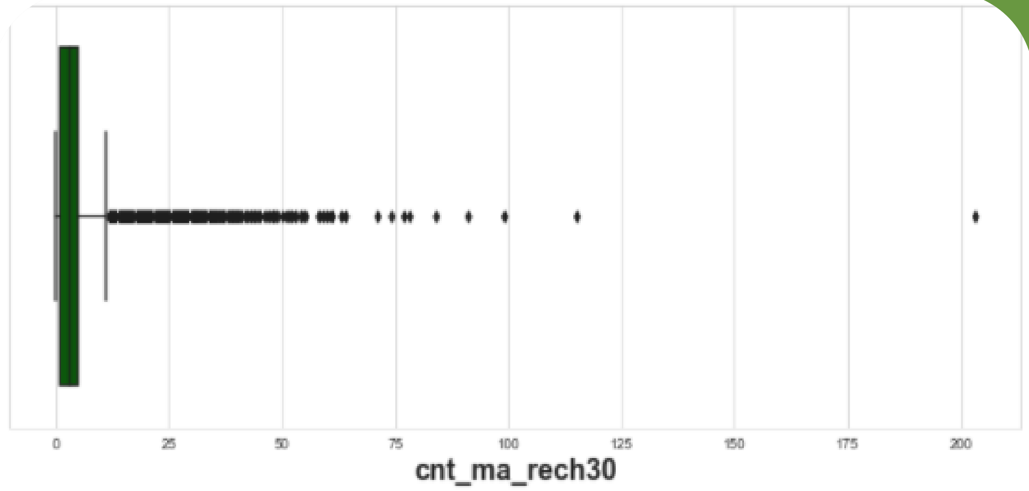


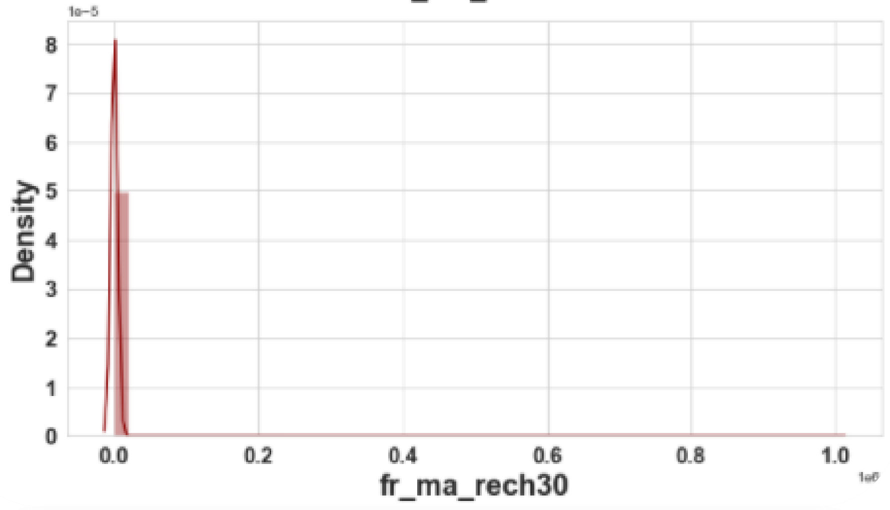
Visualization of count plot & box plot

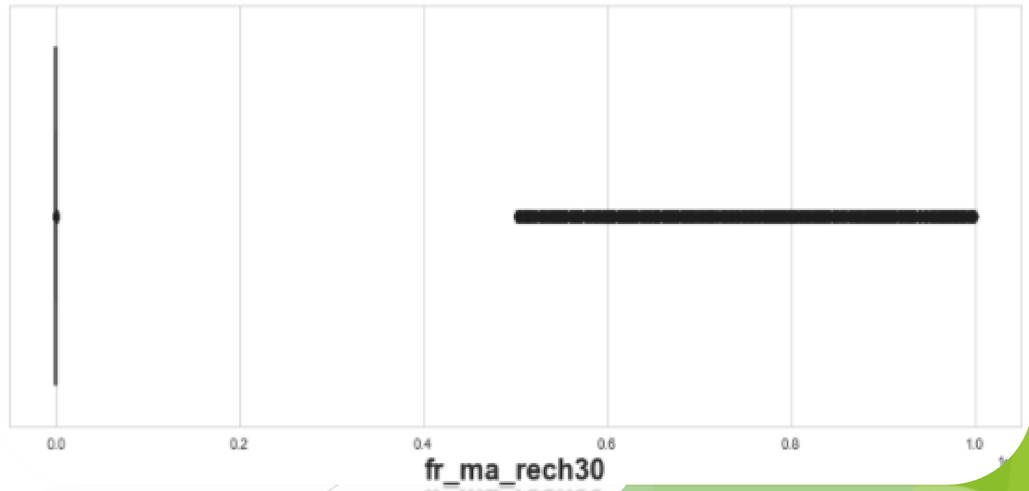
Distribution plot for “cnt_ma_rech30” & “fr_ma_rech30”.

box plot for “cnt_ma_rech30” & “fr_ma_rech30”.



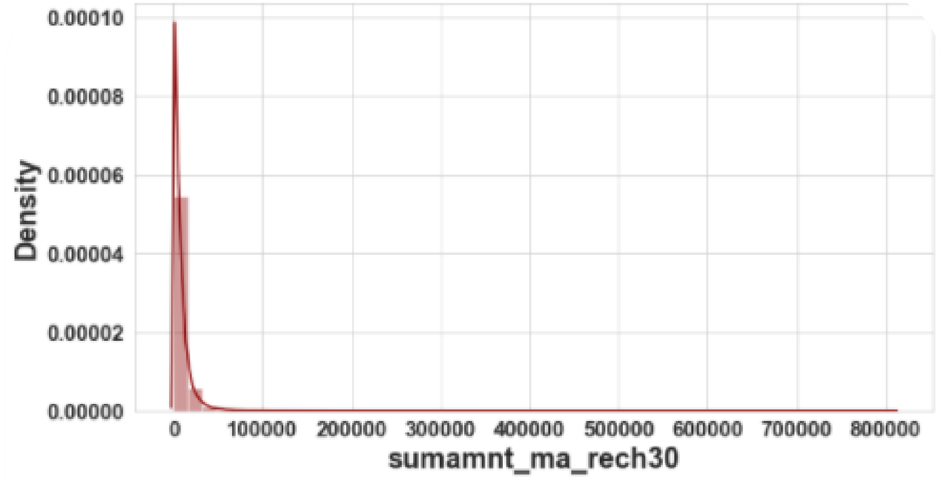


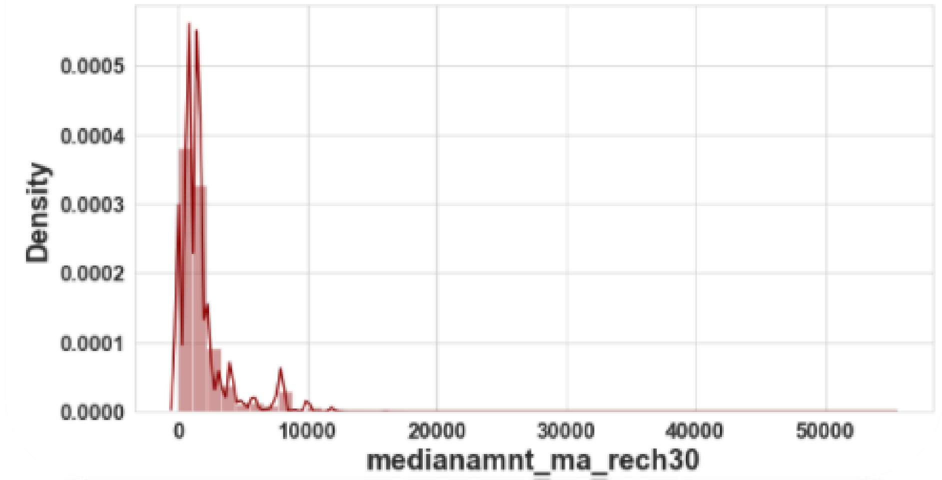




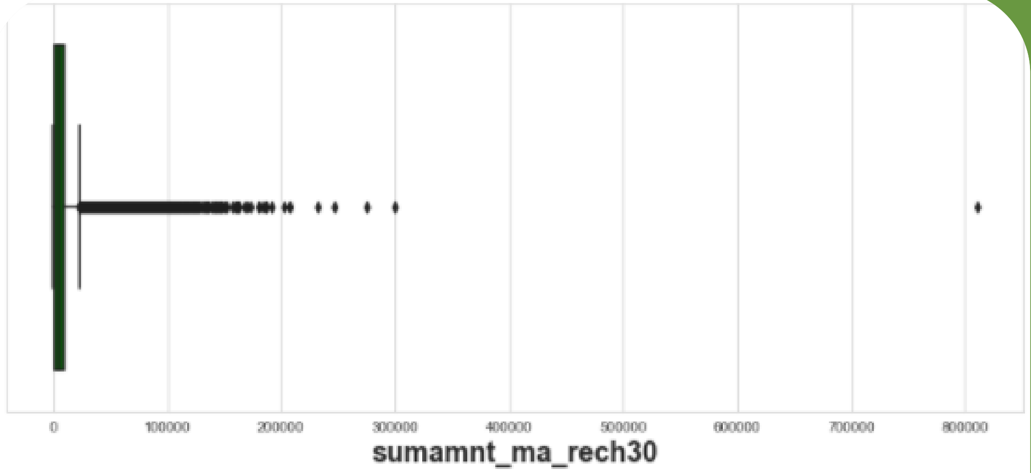
Visualization of count plot & box plot

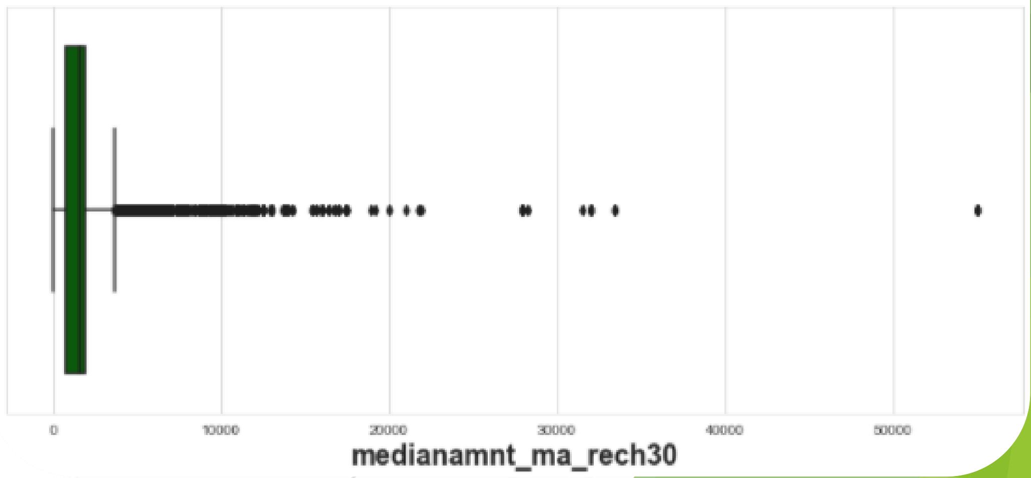
Distribution plot for “sumamnt_ma_rech30” & “medianamnt_ma_rech30”.





box plot for “sumamnt_ma_rech30” & “medianamnt_ma_rech30”.

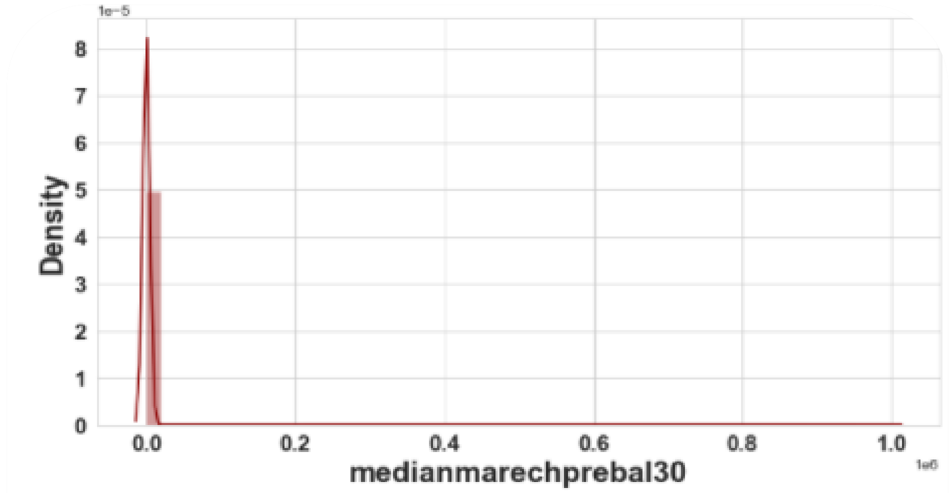


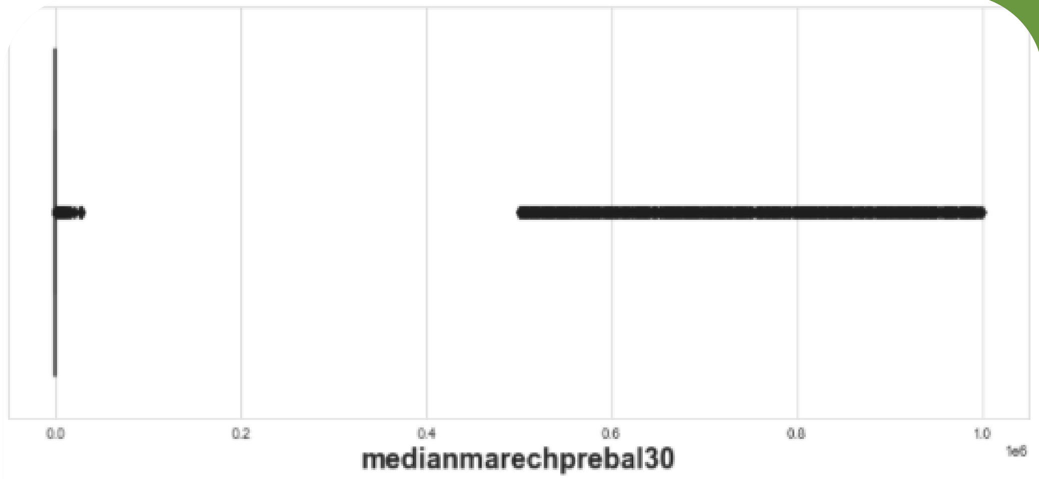


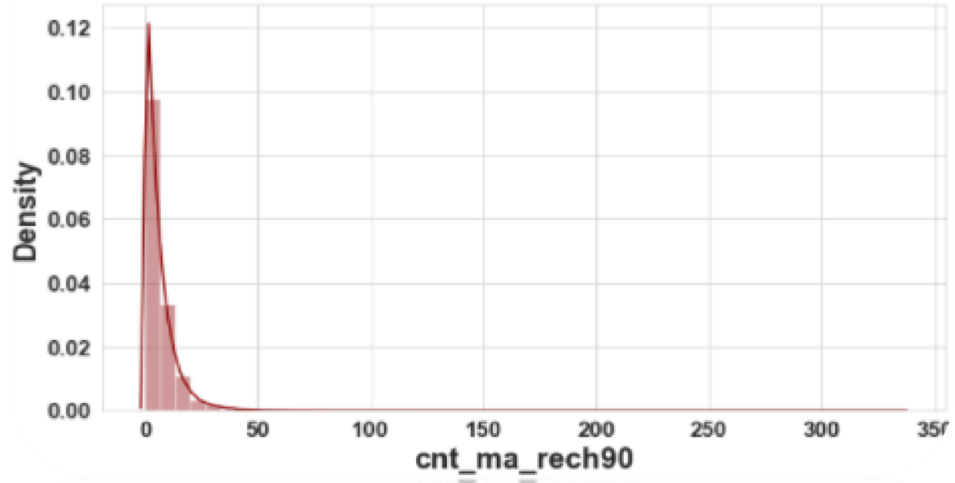
Visualization of count plot & box plot

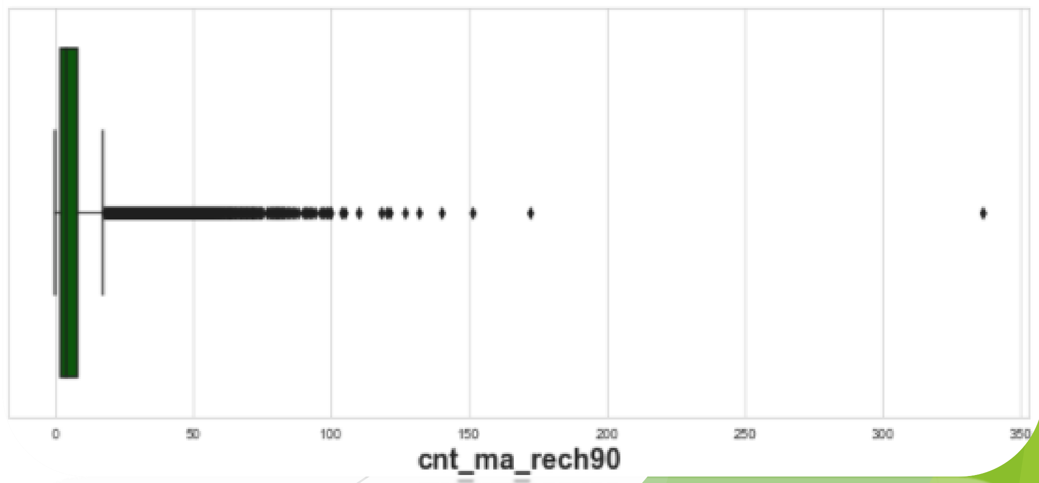
Distribution plot for "medianmarechprebal30" & "cnt_ma_rech90".

box plot for "medianmarechprebal30" & "medianamnt_ma_rech30".





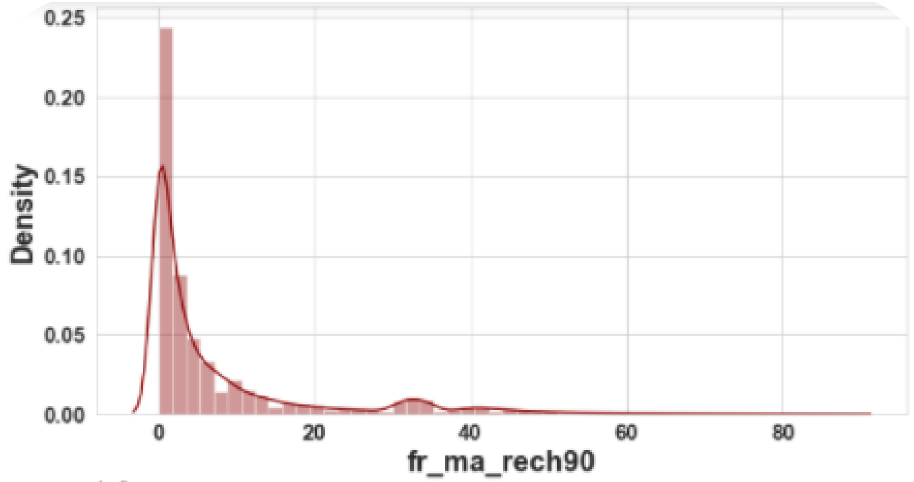


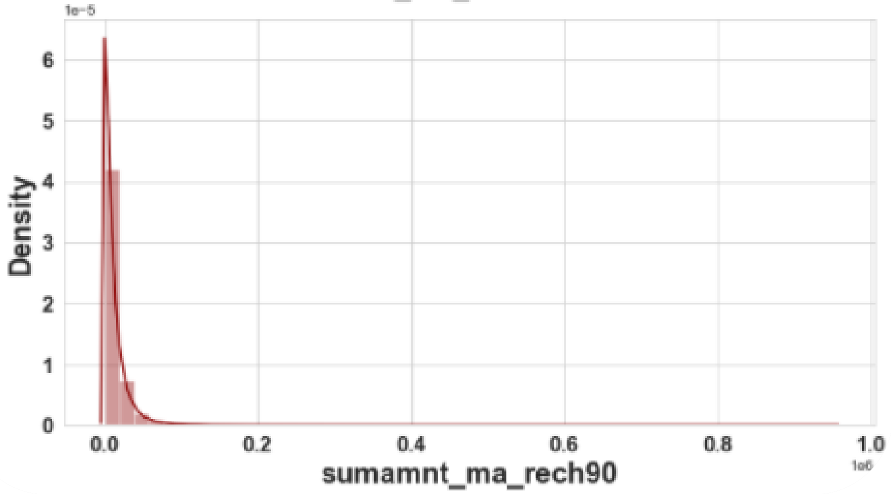


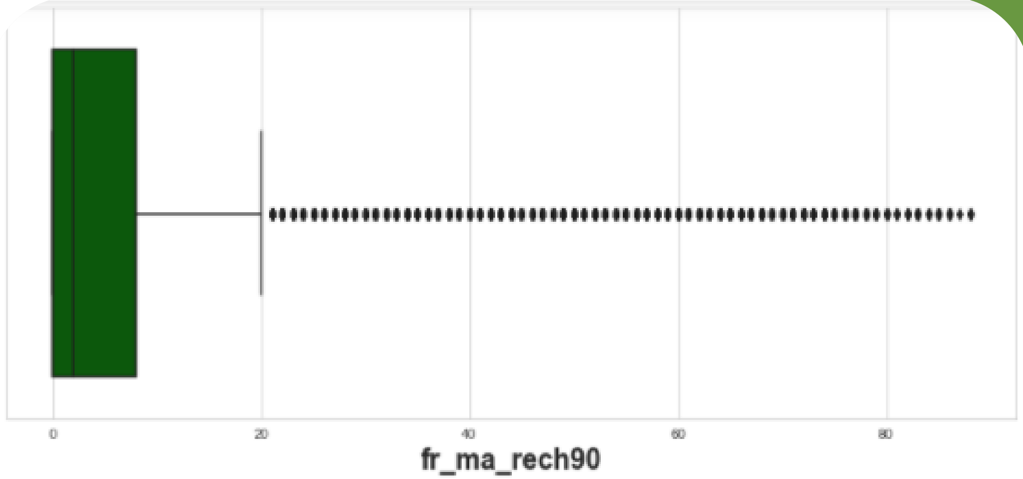
Visualization of count plot & box plot

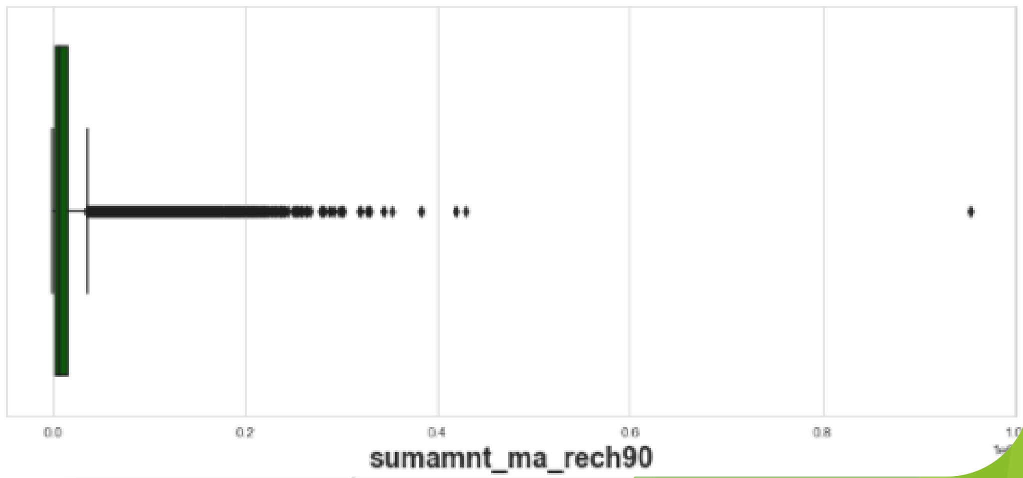
Distribution plot for “fr_ma_rech90” & “sumamnt_ma_rech90”.

box plot for “fr_ma_rech90” & “sumamnt_ma_rech90”.



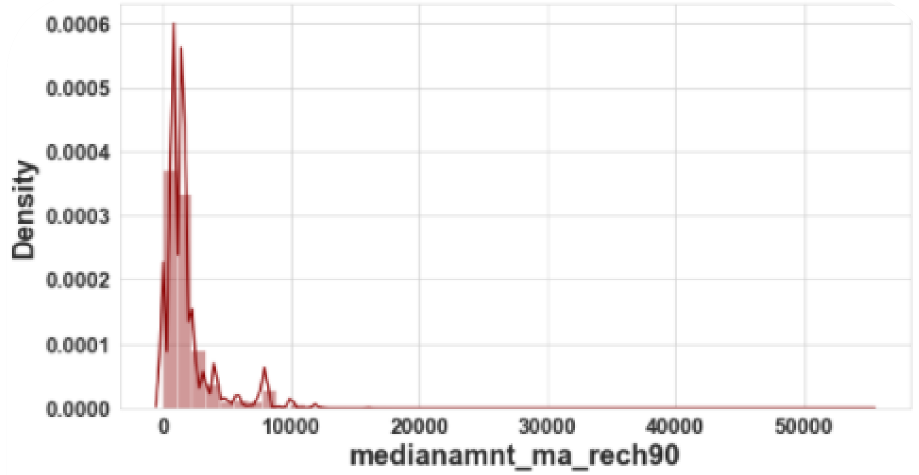


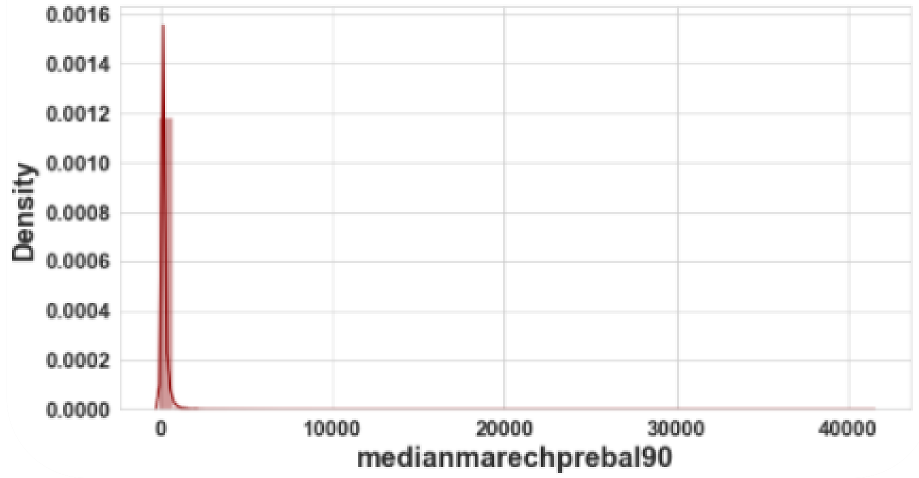




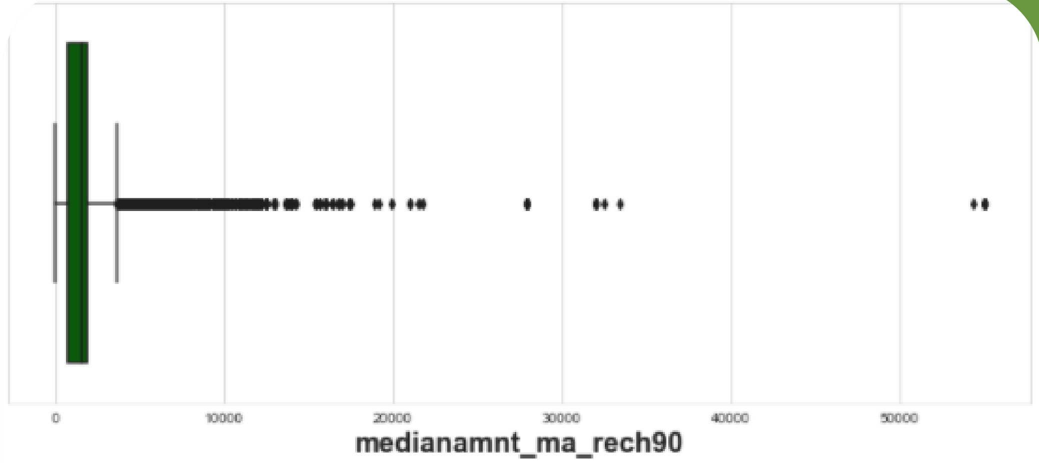
Visualization of count plot & box plot

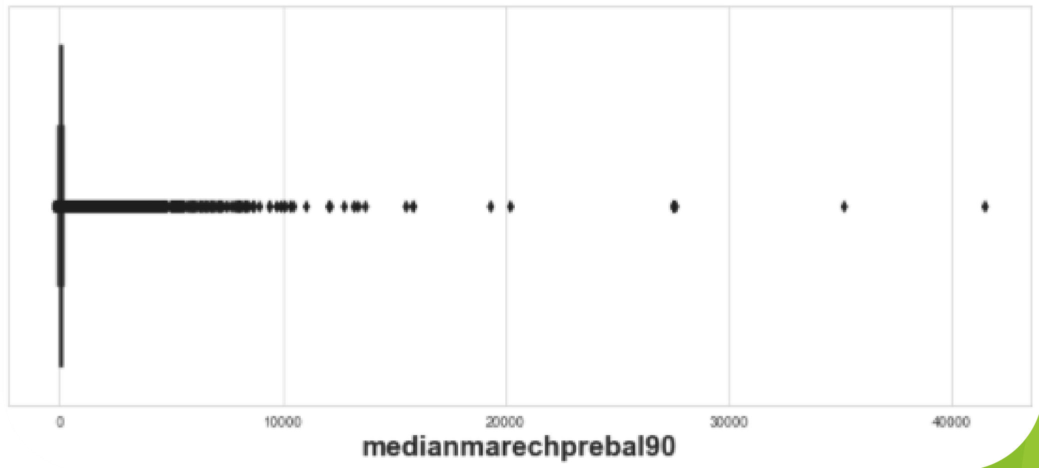
Distribution plot for "medianamnt_ma_rech90" & "medianmarechprebal90".





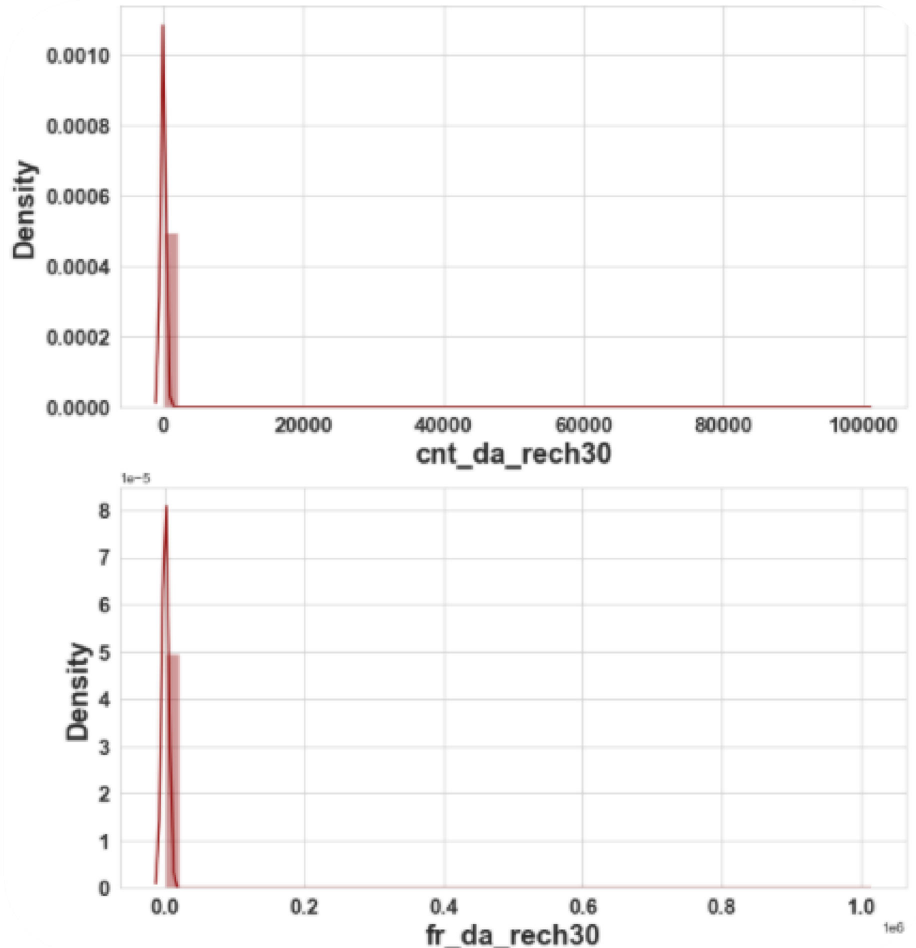
box plot for "medianamnt_ma_rech90" & "medianmarechprebal90".



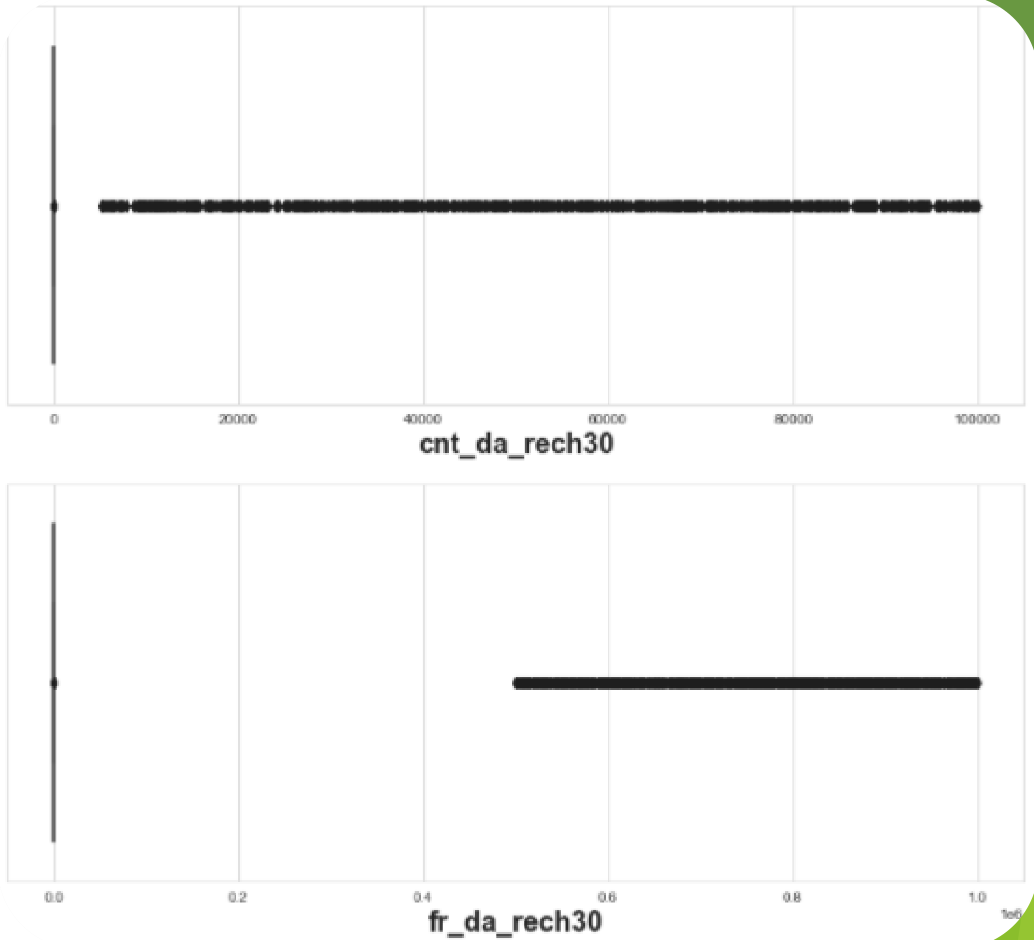


Visualization of count plot & box plot

Distribution plot for “cnt_da_rech30” & “fr_da_rech30”.



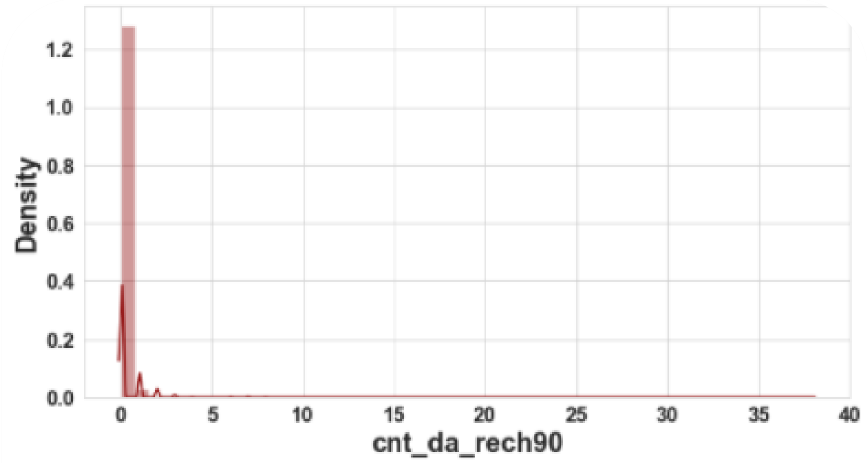
box plot for “cnt_da_rech30” & “fr_da_rech30”.

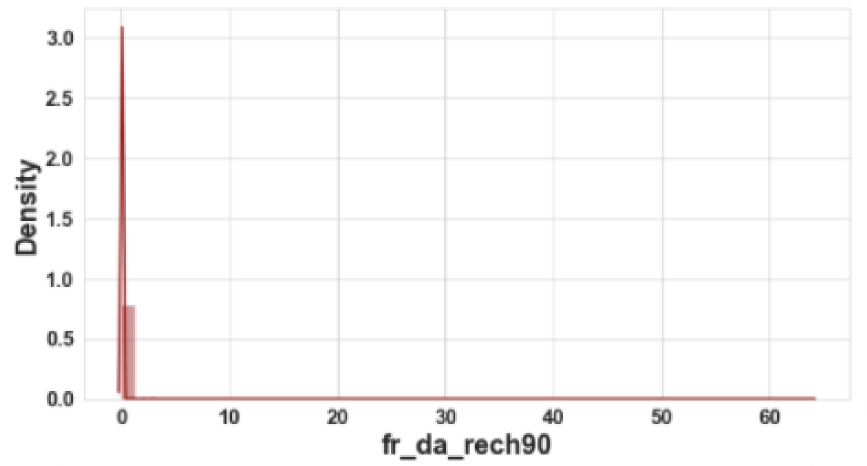


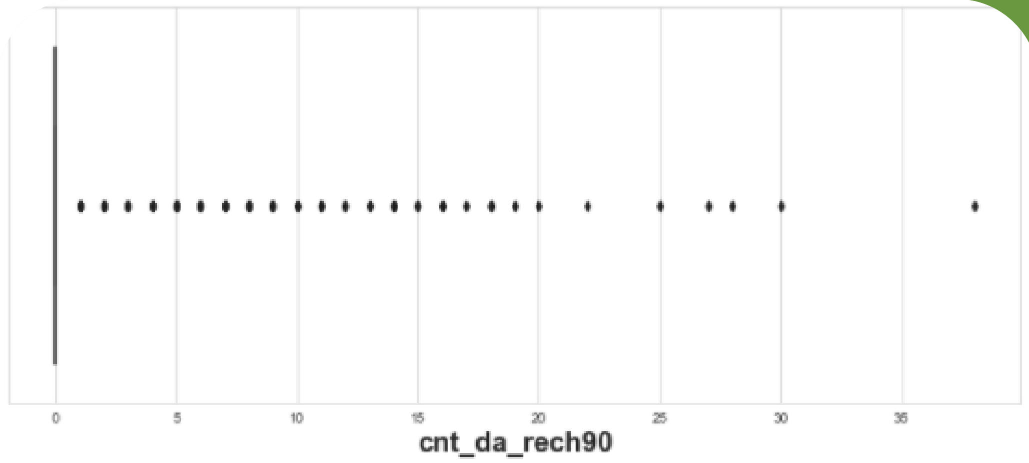
Visualization of count plot & box plot

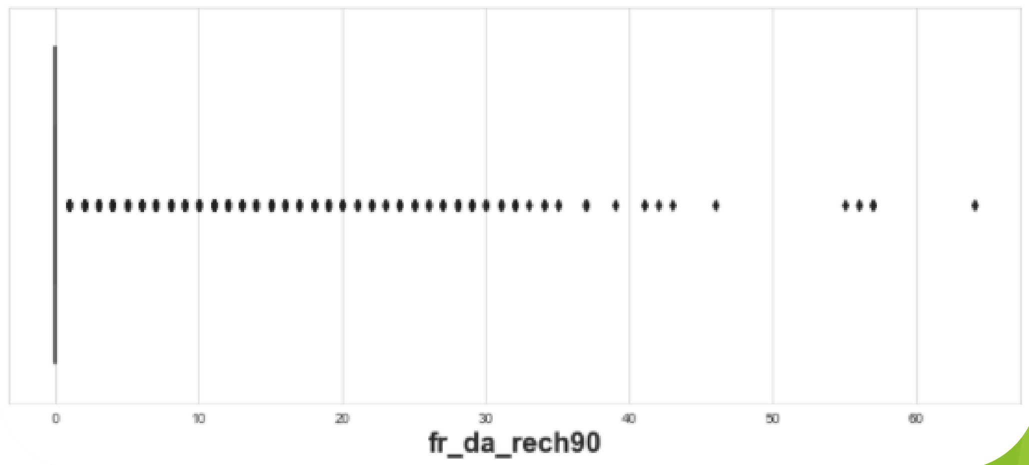
Distribution plot for “cnt_da_rech90” & “fr_da_rech90”.

box plot for “cnt_da_rech90” & “fr_da_rech90”.





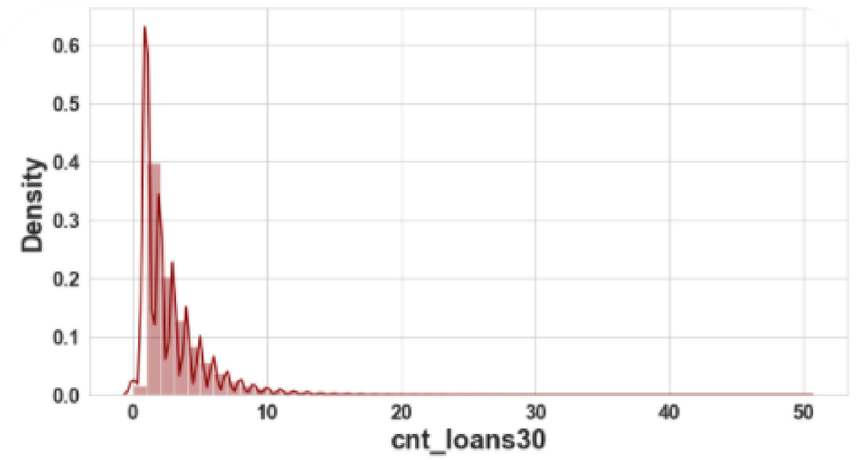


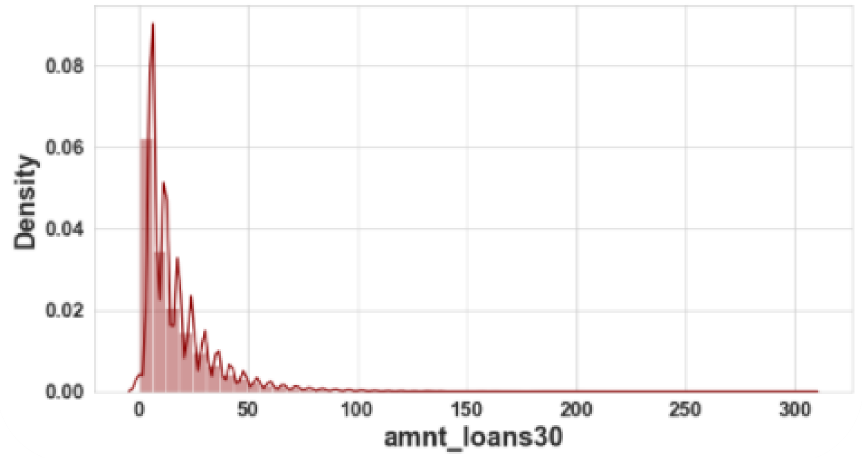


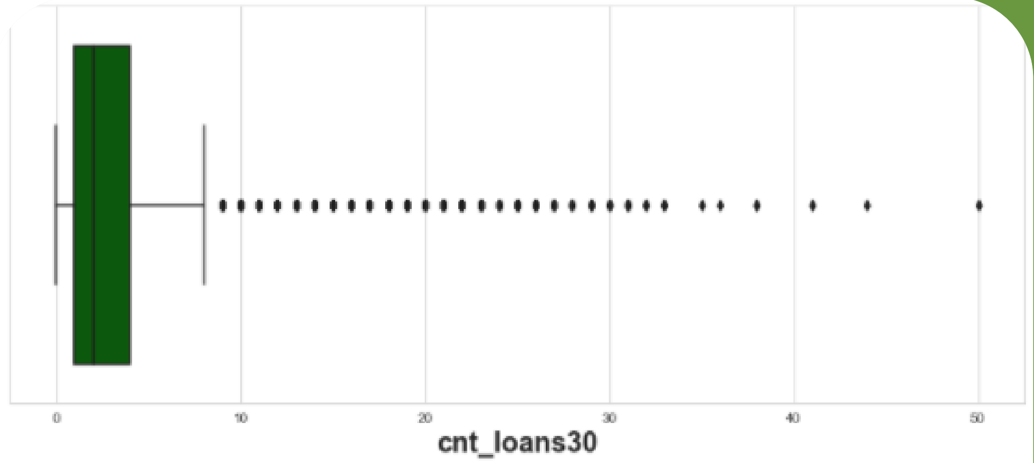
Visualization of count plot & box plot

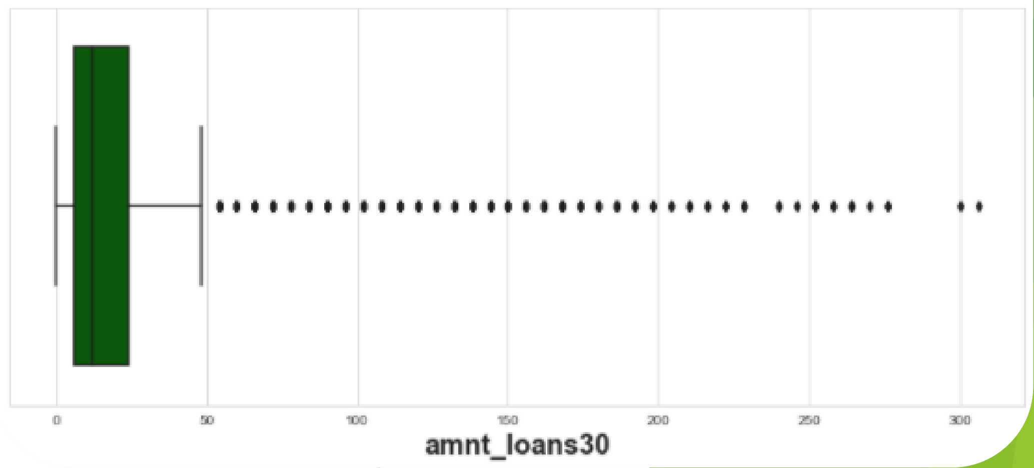
Distribution plot for “cnt_loans30” & “amnt_loans30”.

box plot for “cnt_loans30” & “amnt_loans30”.





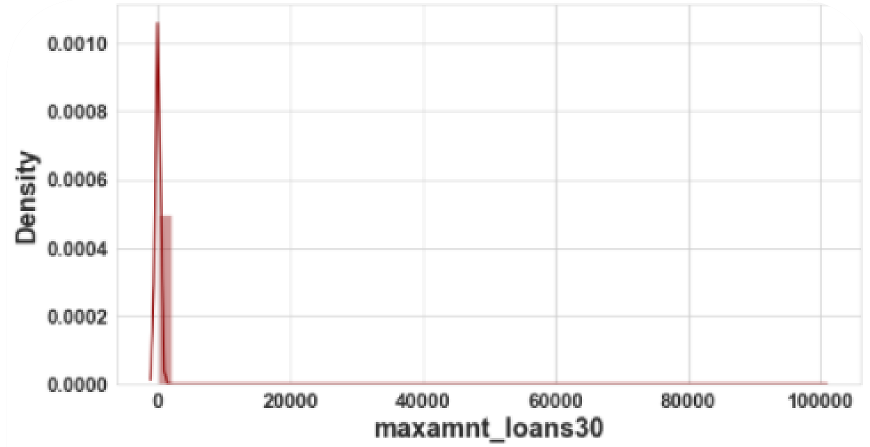


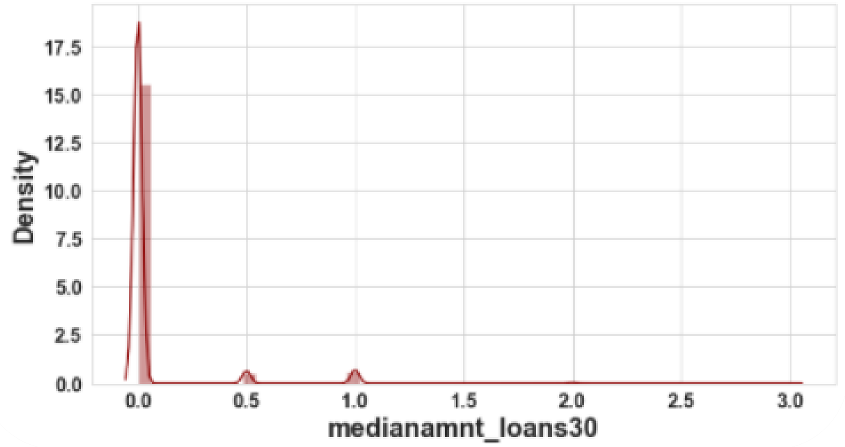


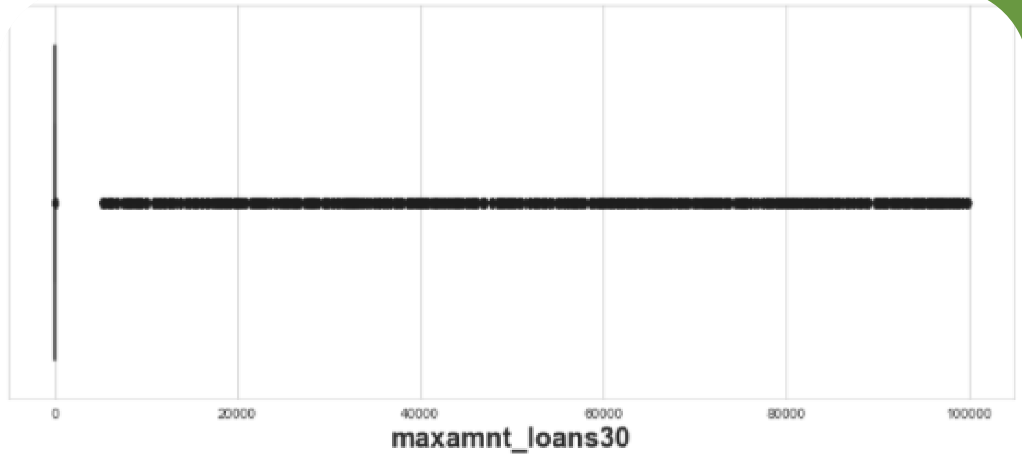
Visualization of count plot & box plot

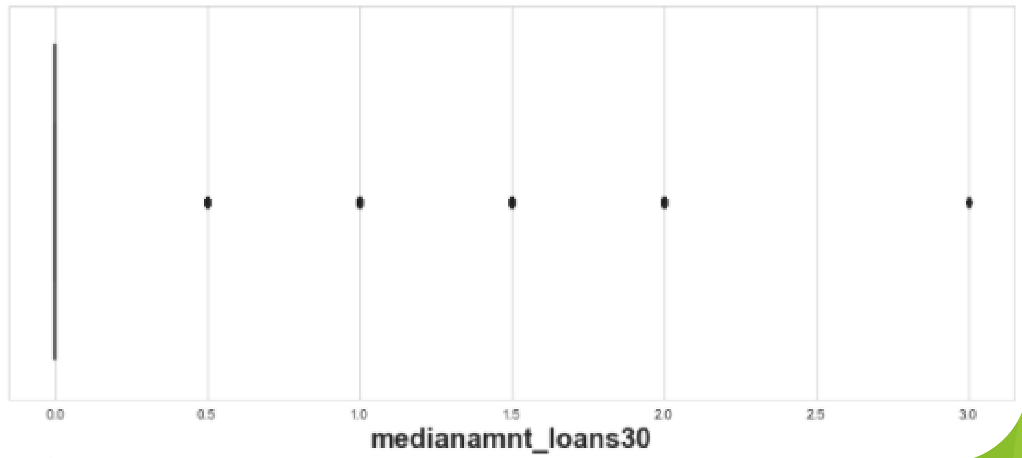
Distribution plot for “maxamnt_loans30” & “medianamnt_loans30”.

box plot for “maxamnt_loans30” & “medianamnt_loans30”.



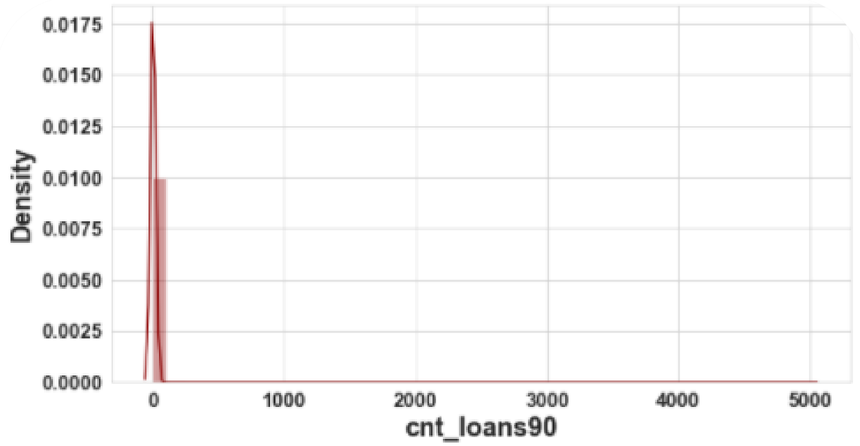


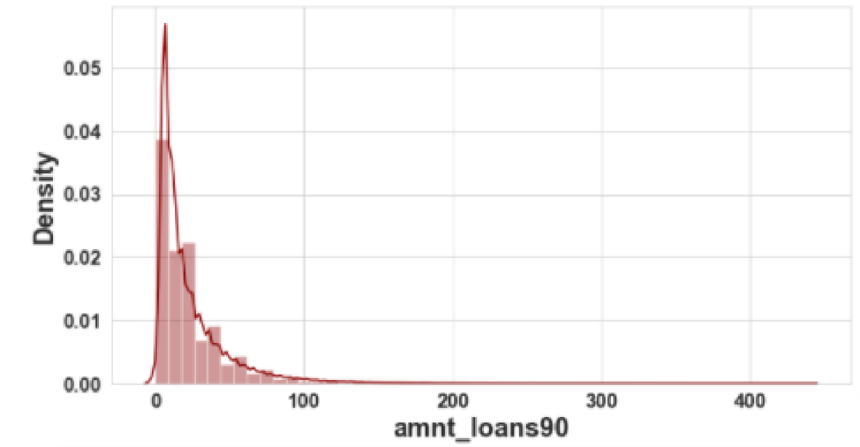




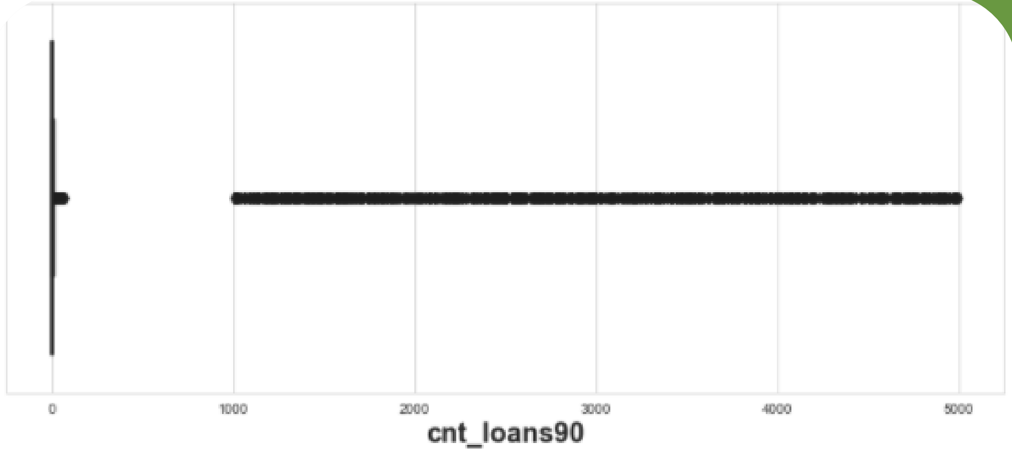
Visualization of count plot & box plot

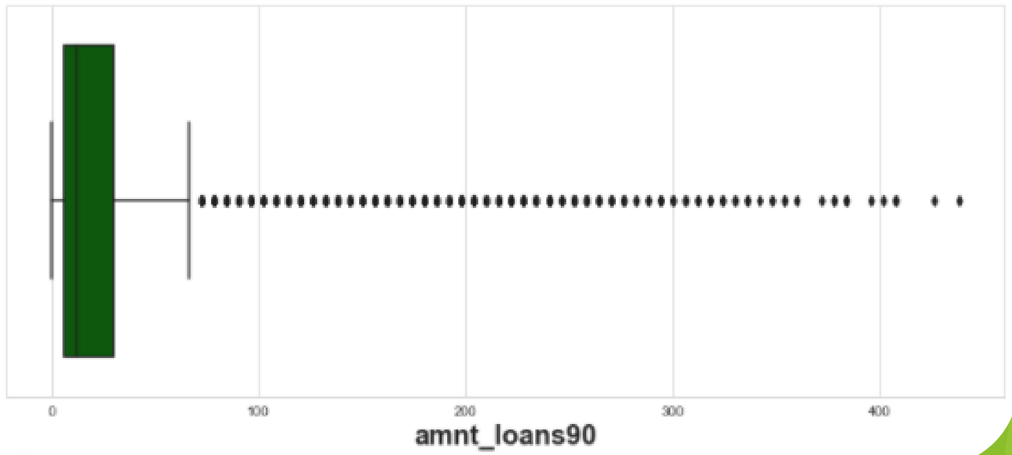
Distribution plot for “cnt_loans90” & “amnt_loans90”.





box plot for “cnt_loans90” & “amnt_loans90”.

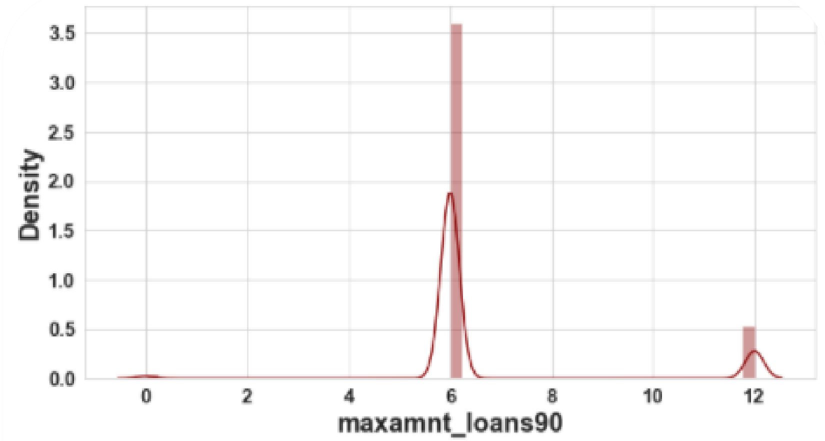


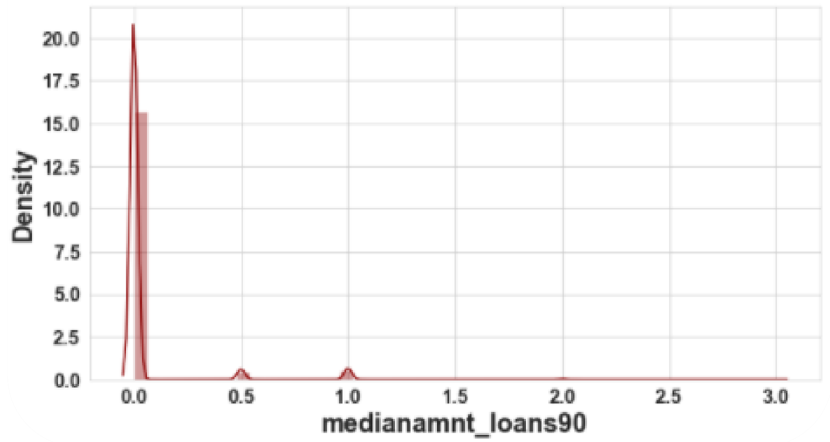


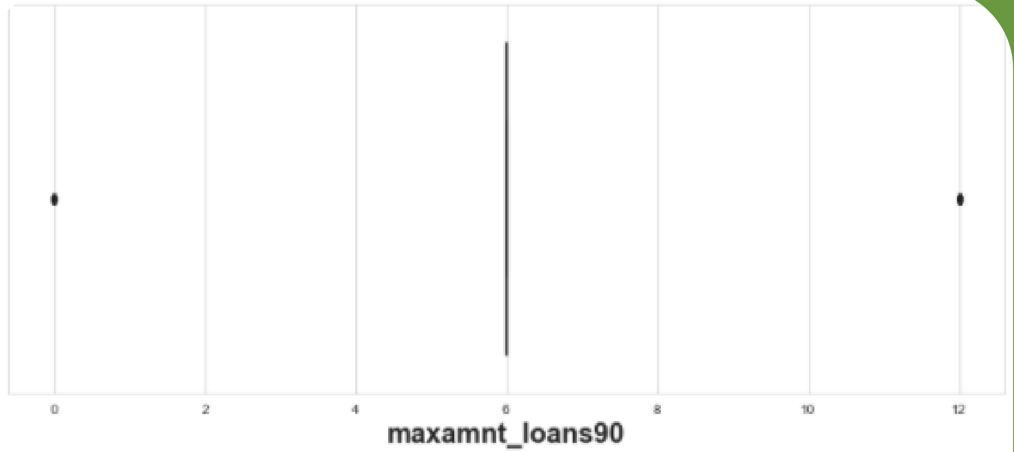
Visualization of count plot & box plot

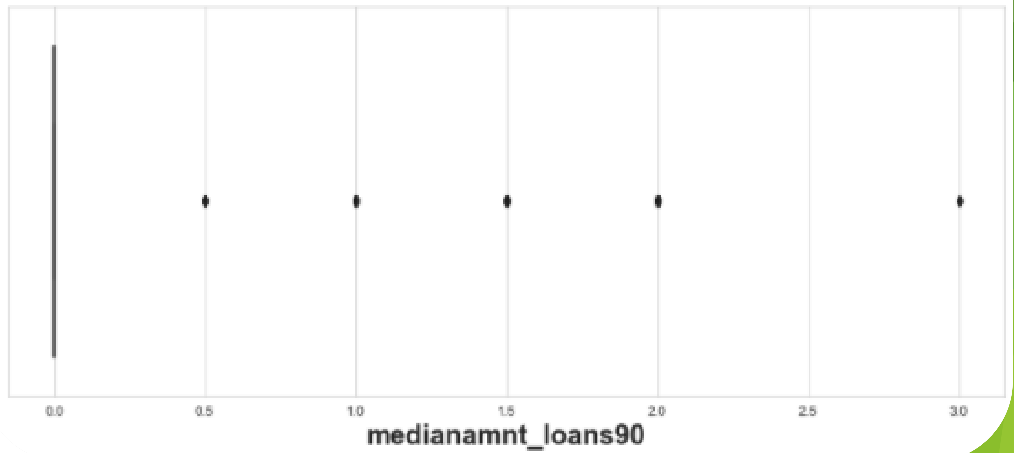
Distribution plot for “maxamnt_loans90” & “medianamntloans90”.

box plot for “maxamnt_loans90” & “medianamntloans90”.



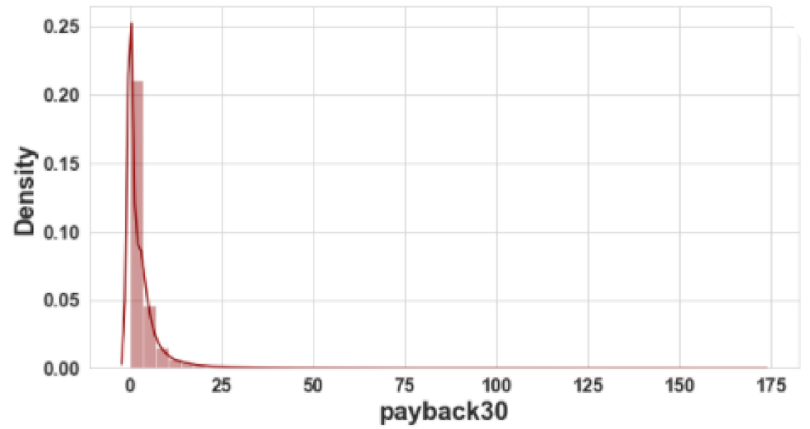


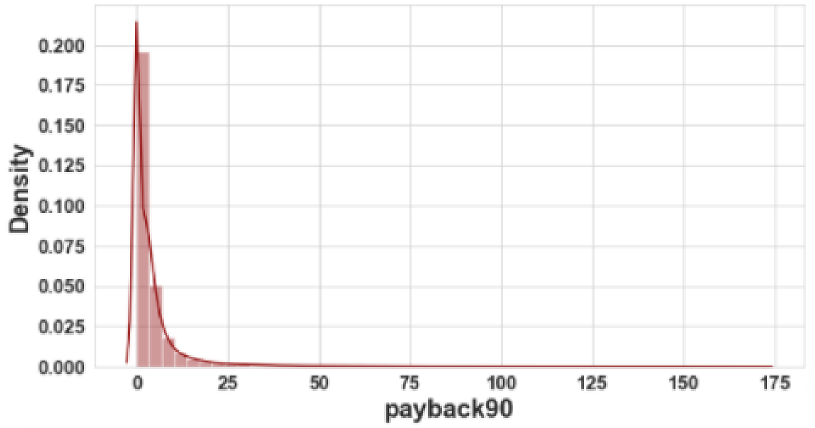




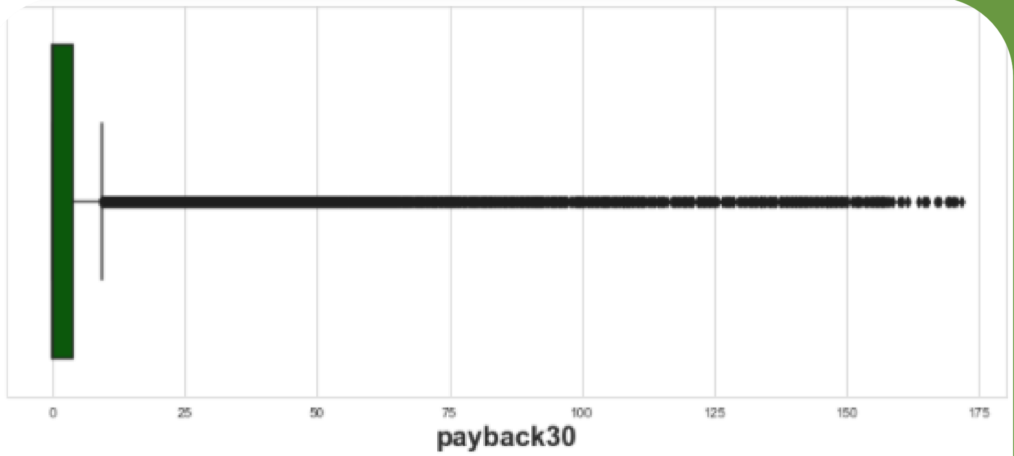
Visualization of count plot & box plot

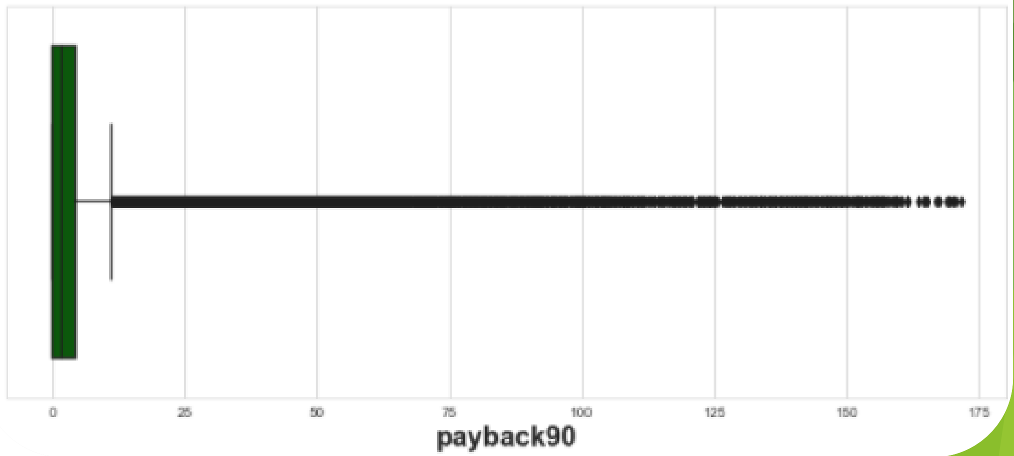
Distribution plot for “payback30” & “payback90”.





box plot for “payback30” & “payback90”.

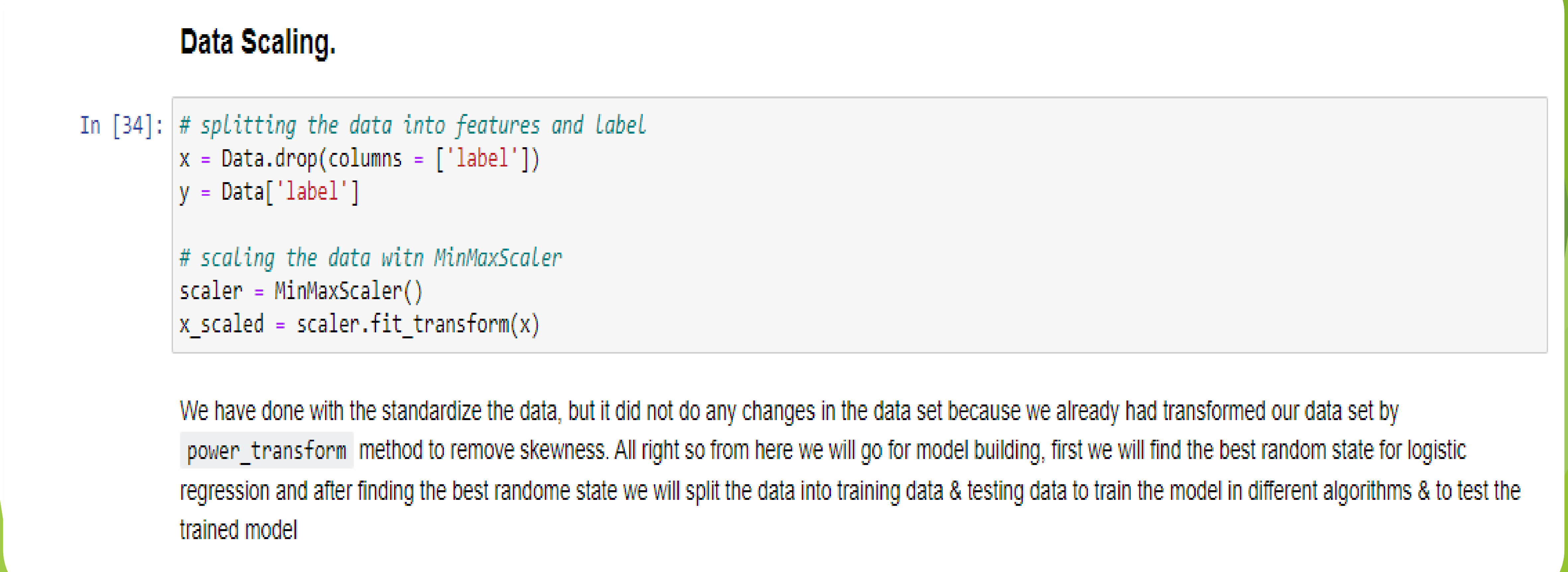




Findings, assumptions, conclusions & next step on the basis of data analysis.

After doing the research on data analysis we found that data was messed up too much, before visualizing we have seen the statistical description of the continuous columns and found that in almost each case standard deviation was greater than the mean of the data the reason for this was because in each case up to 90% data was distributed on either a single point or in a definite range but 10% data was far ahead than them. Box visualization shows us that the amount of the outliers are very much, we can’t afford to loose this much data hence we applied power transformation (yeo-Johnson) method to remove the skewness. It did well in most of the cases but not in all the cases. Because we have transformed the data through power transformation we move ahead in the process of model building. After this we did some preprocessing of the data, we scaled the data, found the best random state for train test & split, than we split the data into train data & test data. After preprocessing we started building the model.

Data scaling code & documentation.

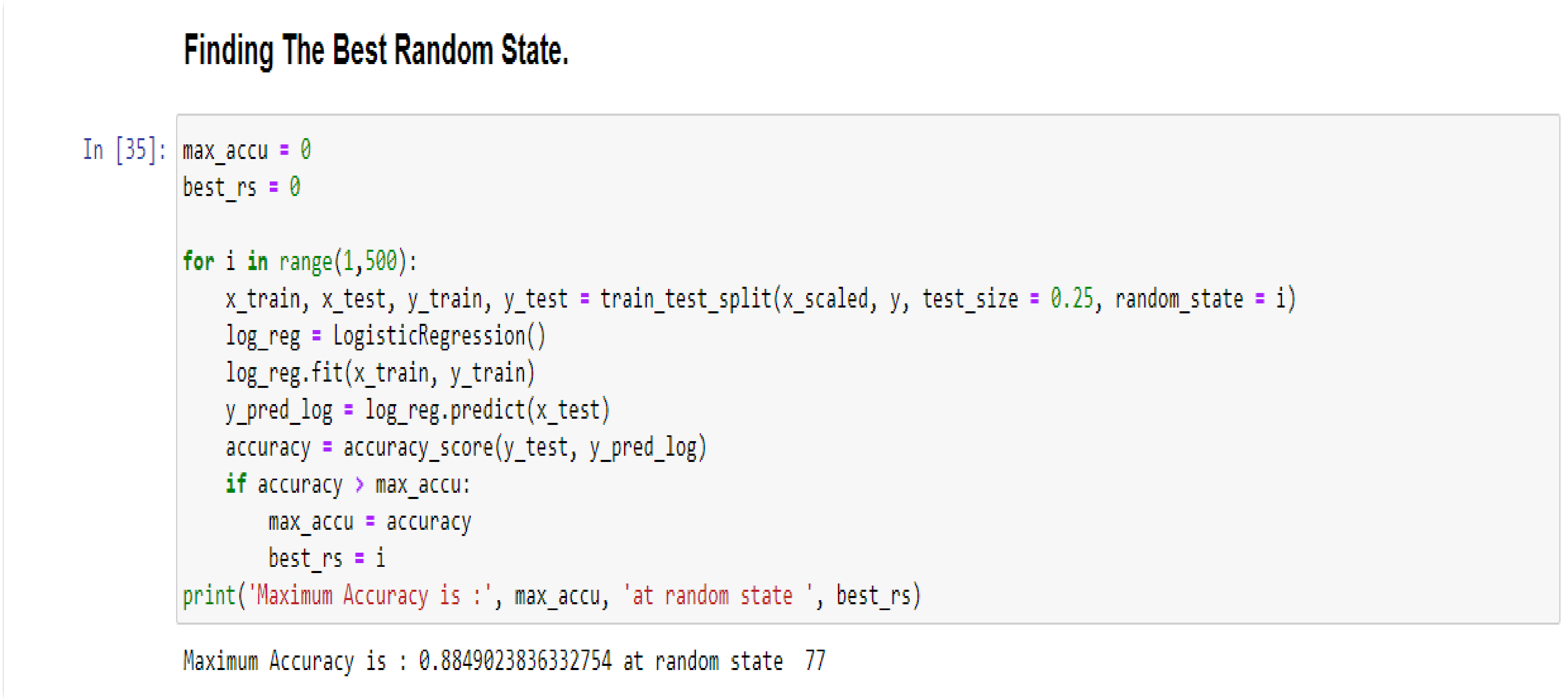


Data Scaling.

```
In [34]: # splitting the data into features and label
         x = Data.drop(columns = ['label'])
         y = Data['label']

         # scaling the data witn MinMaxScaler
         scaler = MinMaxScaler()
         x_scaled = scaler.fit_transform(x)
```

We have done with the standardize the data, but it did not do any changes in the data set because we already had transformed our data set by `power_transform` method to remove skewness. All right so from here we will go for model building, first we will find the best random state for logistic regression and after finding the best randome state we will split the data into training data & testing data to train the model in different algorithms & to test the trained model

Finding of best random state for train & test data.



Finding The Best Random State.

```
In [35]: max_accu = 0
         best_rs = 0

         for i in range(1,500):
             x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size = 0.25, random_state = i)
             log_reg = LogisticRegression()
             log_reg.fit(x_train, y_train)
             y_pred_log = log_reg.predict(x_test)
             accuracy = accuracy_score(y_test, y_pred_log)
             if accuracy > max_accu:
                 max_accu = accuracy
                 best_rs = i
         print('Maximum Accuracy is :', max_accu, 'at random state ', best_rs)
```

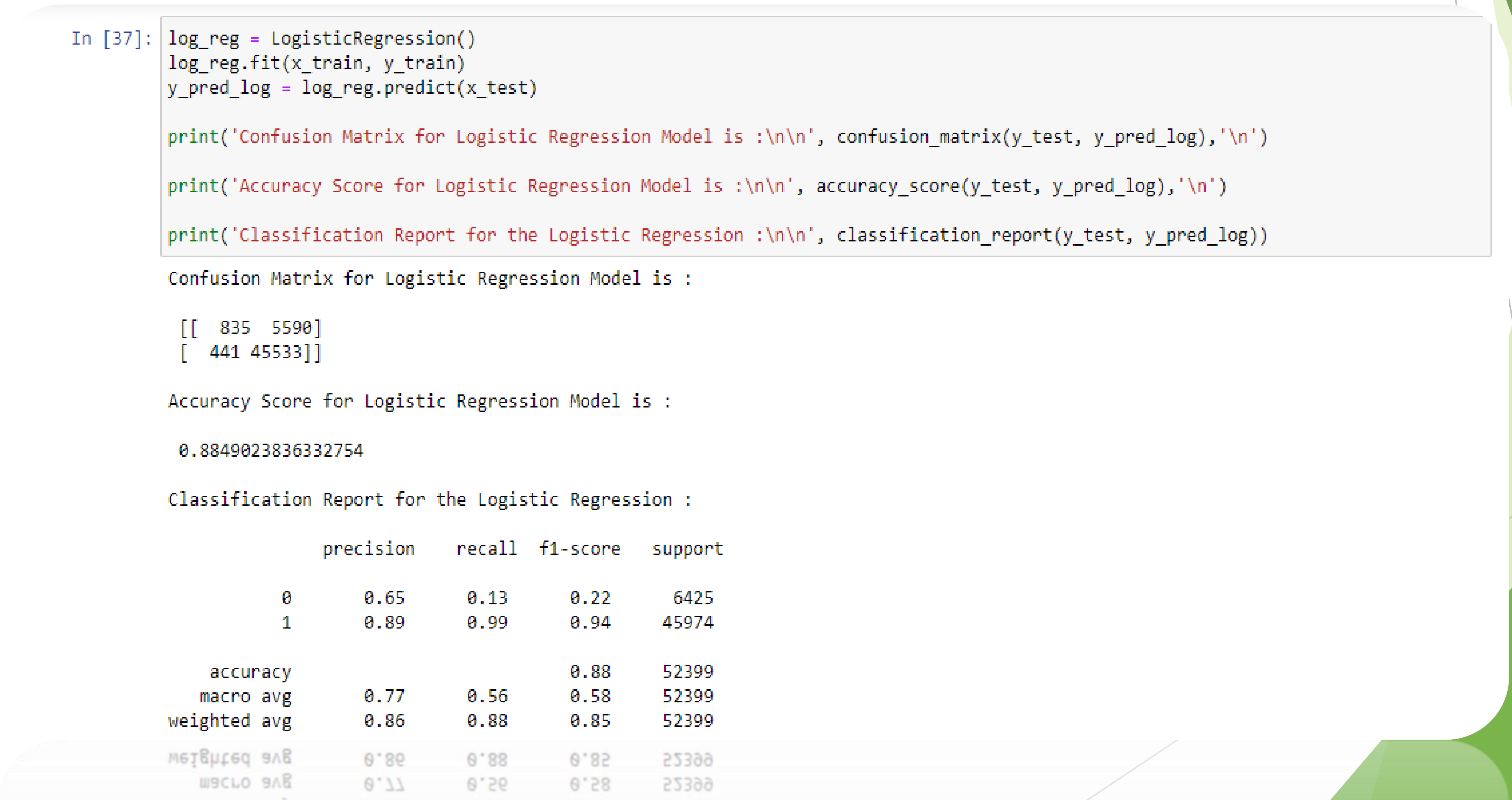
```
Maximum Accuracy is : 0.8849023836332754 at random state  77
```

Machine learning model building

After doing the preprocessing like data scaling, splitting the data into train data & test data and finding the best random state we started doing machine learning model building with the help of scikit learn library. We have chosen to build 5 models for the project.

- Logistic Regression.
- K-Nearest Neighbors.
- Decision Tree.
- Random Forest.
- Support Vector Machine.

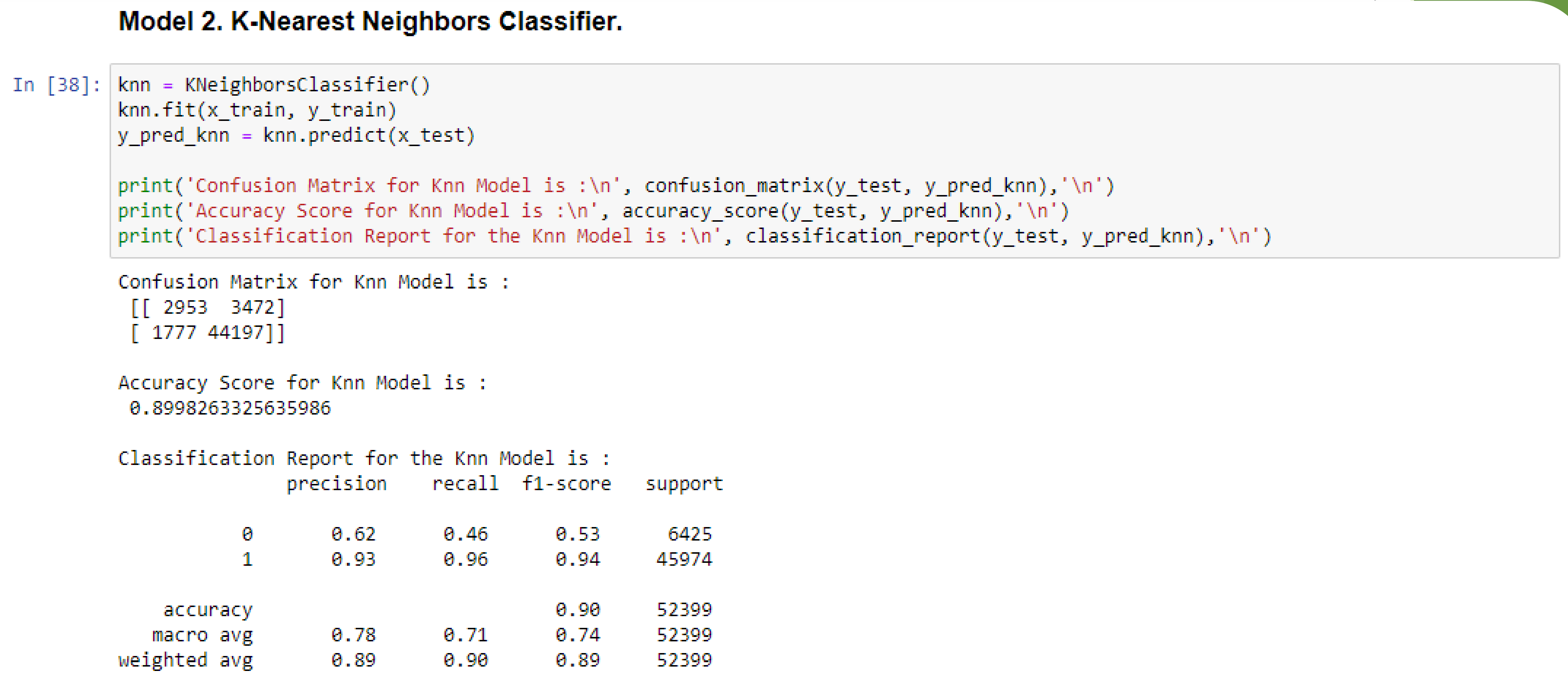
Logistic regression model code & metrics evaluations.



```
In [37]: log_reg = LogisticRegression()
         log_reg.fit(x_train, y_train)
         y_pred_log = log_reg.predict(x_test)

         print('Confusion Matrix for Logistic Regression Model is :\n\n', confusion_matrix(y_test, y_pred_log),'\n')

         print('Accuracy Score for Logistic Regression Model is :\n\n', accuracy_score(y_test, y_pred_log),'\n')

         print('Classification Report for the Logistic Regression :\n\n', classification_report(y_test, y_pred_log))
```

```
Confusion Matrix for Logistic Regression Model is :

 [[  835  5590]
 [  441 45533]]

Accuracy Score for Logistic Regression Model is :

 0.8849023836332754

Classification Report for the Logistic Regression :

               precision    recall  f1-score   support

           0       0.65      0.13      0.22      6425
           1       0.89      0.99      0.94     45974

    accuracy                           0.88     52399
   macro avg       0.77      0.56      0.58     52399
weighted avg       0.86      0.88      0.85     52399
```

Knn model code & metrics evaluations.

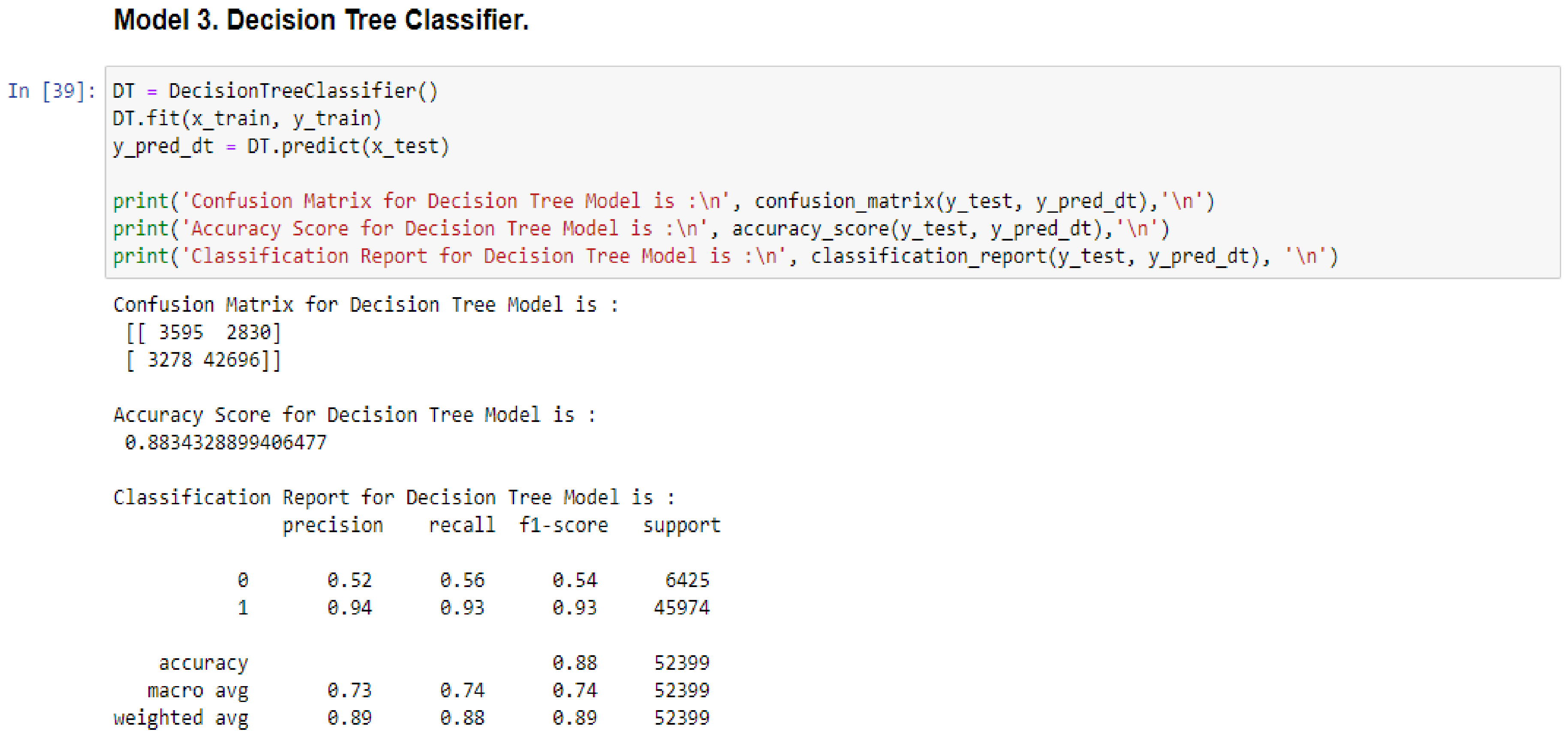


Model 2. K-Nearest Neighbors Classifier.

```
In [38]: knn = KNeighborsClassifier()
         knn.fit(x_train, y_train)
         y_pred_knn = knn.predict(x_test)

         print('Confusion Matrix for Knn Model is :\n', confusion_matrix(y_test, y_pred_knn),'\n')
         print('Accuracy Score for Knn Model is :\n', accuracy_score(y_test, y_pred_knn),'\n')
         print('Classification Report for the Knn Model is :\n', classification_report(y_test, y_pred_knn),'\n')
```

```
Confusion Matrix for Knn Model is :
 [[ 2953  3472]
 [ 1777 44197]]

Accuracy Score for Knn Model is :
 0.8998263325635986

Classification Report for the Knn Model is :
               precision    recall  f1-score   support

           0       0.62      0.46      0.53      6425
           1       0.93      0.96      0.94     45974

    accuracy                           0.90     52399
   macro avg       0.78      0.71      0.74     52399
weighted avg       0.89      0.90      0.89     52399
```

Decision tree model code & metrics evaluations.

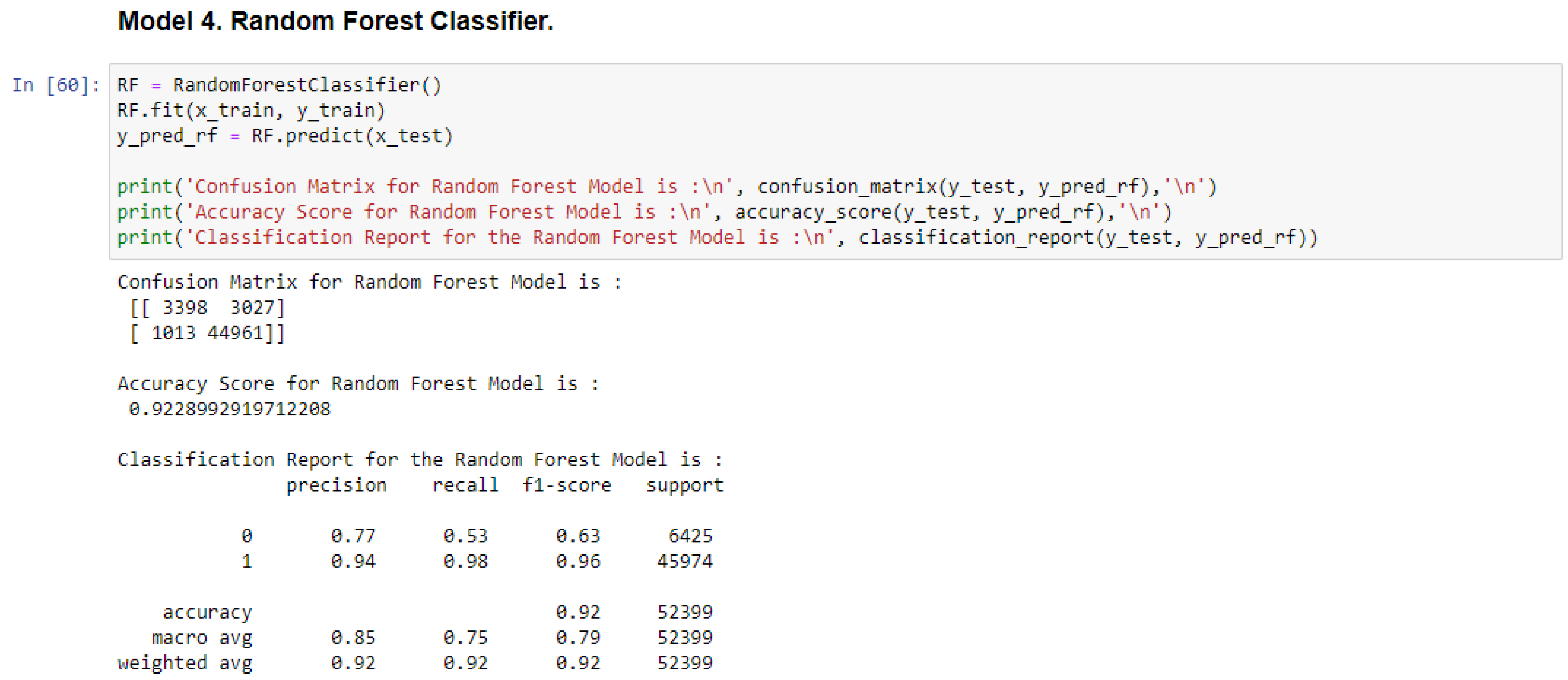


Model 3. Decision Tree Classifier.

```
In [39]: DT = DecisionTreeClassifier()
         DT.fit(x_train, y_train)
         y_pred_dt = DT.predict(x_test)

         print('Confusion Matrix for Decision Tree Model is :\n', confusion_matrix(y_test, y_pred_dt),'\n')
         print('Accuracy Score for Decision Tree Model is :\n', accuracy_score(y_test, y_pred_dt),'\n')
         print('Classification Report for Decision Tree Model is :\n', classification_report(y_test, y_pred_dt), '\n')
```

```
Confusion Matrix for Decision Tree Model is :
 [[ 3595  2830]
 [ 3278 42696]]

Accuracy Score for Decision Tree Model is :
 0.8834328899406477

Classification Report for Decision Tree Model is :
               precision    recall  f1-score   support

           0       0.52      0.56      0.54      6425
           1       0.94      0.93      0.93     45974

    accuracy                           0.88     52399
   macro avg       0.73      0.74      0.74     52399
weighted avg       0.89      0.88      0.89     52399
```

random forest model code & metrics evaluations.

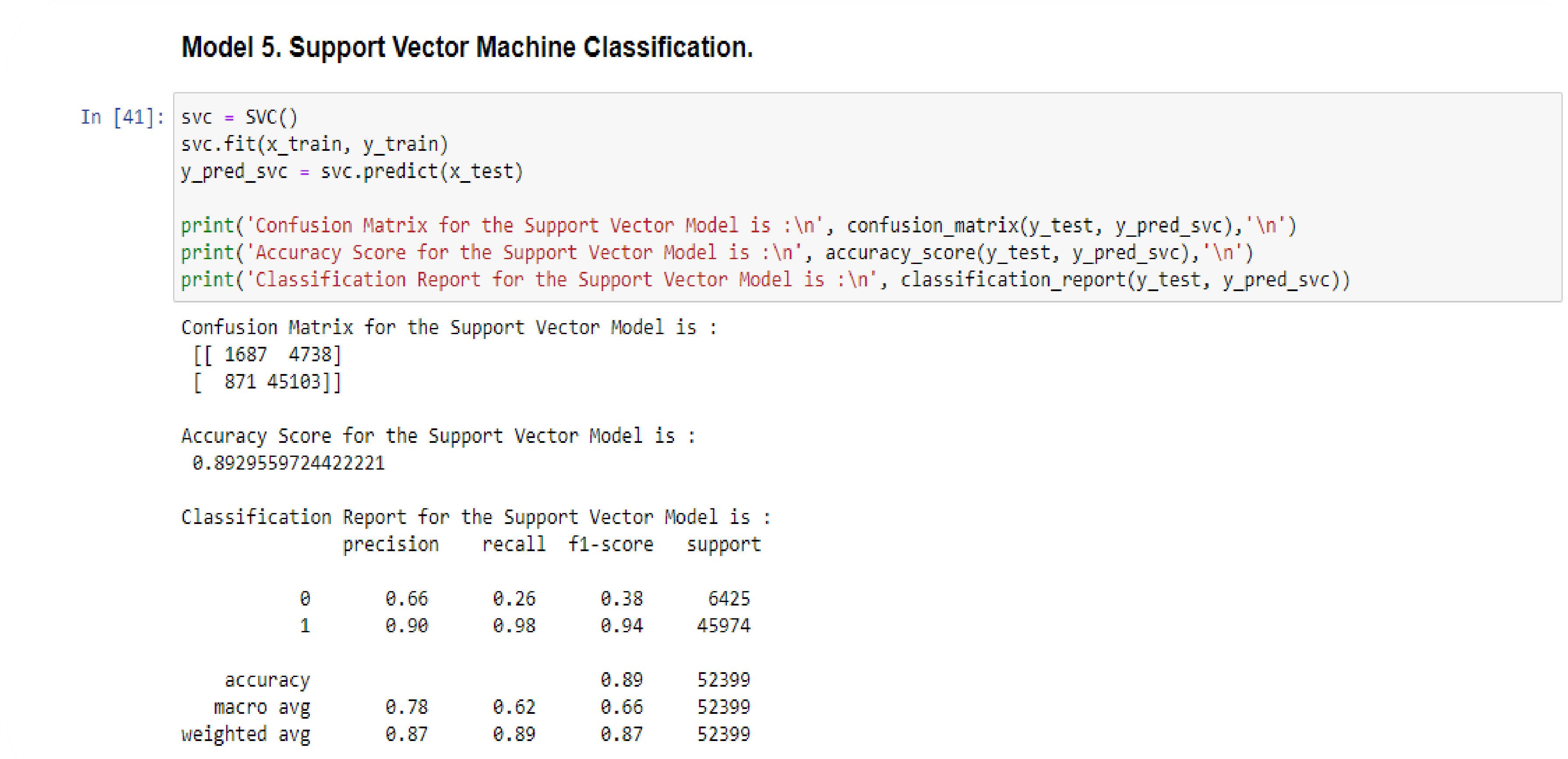


Model 4. Random Forest Classifier.

```
In [60]: RF = RandomForestClassifier()
         RF.fit(x_train, y_train)
         y_pred_rf = RF.predict(x_test)

         print('Confusion Matrix for Random Forest Model is :\n', confusion_matrix(y_test, y_pred_rf),'\n')
         print('Accuracy Score for Random Forest Model is :\n', accuracy_score(y_test, y_pred_rf),'\n')
         print('Classification Report for the Random Forest Model is :\n', classification_report(y_test, y_pred_rf))
```

```
Confusion Matrix for Random Forest Model is :
 [[ 3398  3027]
 [ 1013 44961]]

Accuracy Score for Random Forest Model is :
 0.9228992919712208

Classification Report for the Random Forest Model is :
               precision    recall  f1-score   support

           0       0.77      0.53      0.63      6425
           1       0.94      0.98      0.96     45974

    accuracy                           0.92     52399
   macro avg       0.85      0.75      0.79     52399
weighted avg       0.92      0.92      0.92     52399
```

Support vector machine model code & metrics evaluations.



Model 5. Support Vector Machine Classification.

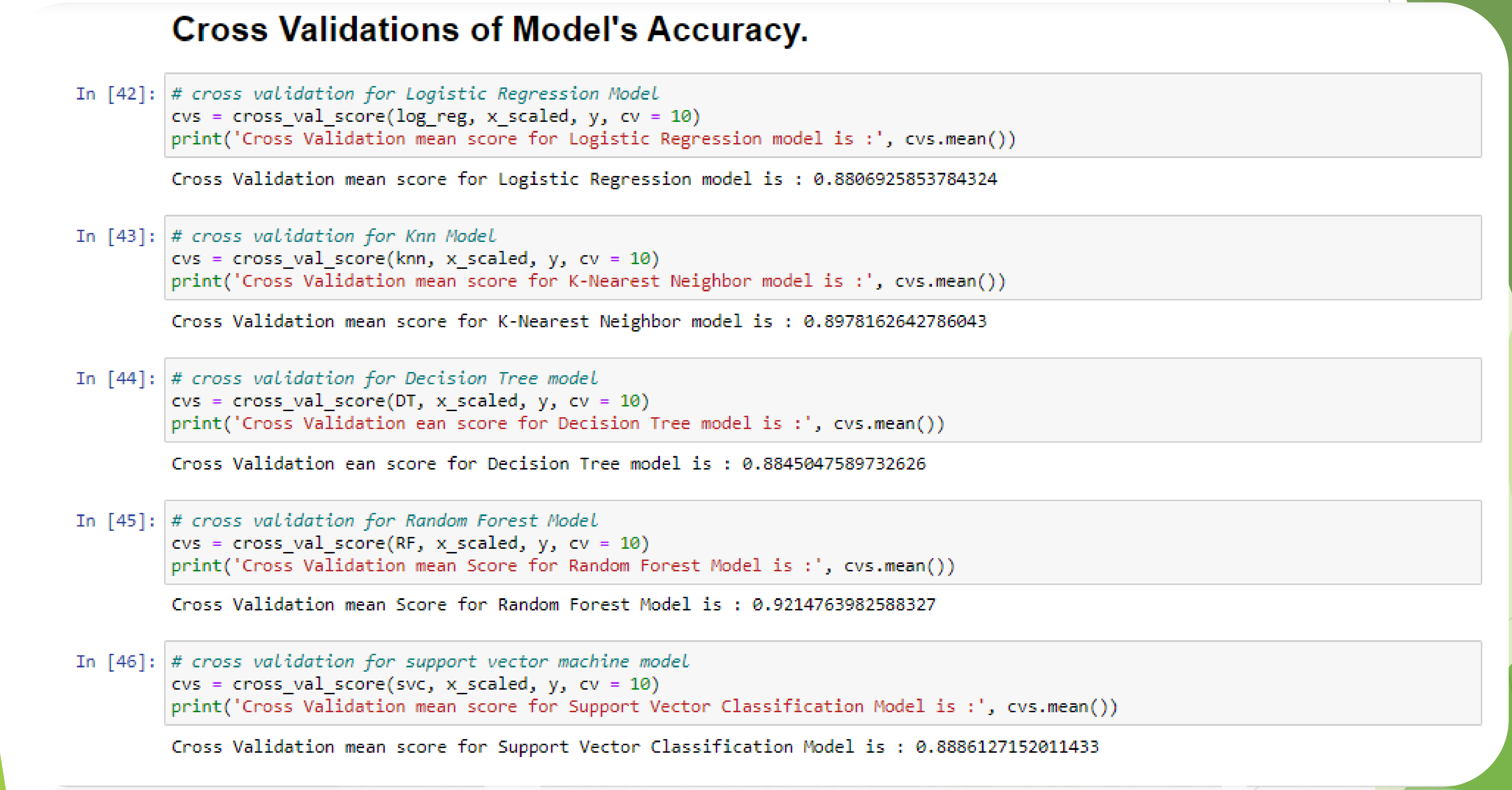
```
In [41]: svc = SVC()
         svc.fit(x_train, y_train)
         y_pred_svc = svc.predict(x_test)

         print('Confusion Matrix for the Support Vector Model is :\n', confusion_matrix(y_test, y_pred_svc),'\n')
         print('Accuracy Score for the Support Vector Model is :\n', accuracy_score(y_test, y_pred_svc),'\n')
         print('Classification Report for the Support Vector Model is :\n', classification_report(y_test, y_pred_svc))
```

```
Confusion Matrix for the Support Vector Model is :
 [[ 1687  4738]
 [  871 45103]]

Accuracy Score for the Support Vector Model is :
 0.8929559724422221

Classification Report for the Support Vector Model is :
               precision    recall  f1-score   support

           0       0.66      0.26      0.38      6425
           1       0.90      0.98      0.94     45974

    accuracy                           0.89     52399
   macro avg       0.78      0.62      0.66     52399
weighted avg       0.87      0.89      0.87     52399
```

Conclusions of machine learning models.

- We build 5 supervised machine learning classification models as we discussed above.
- We got very good accuracy score in all 5 models, the least score was of decision tree model (88.34%) & the maximum score was of Random Forest(92.28).
- But we had very poor precision, recall & f1 score on every model except random forest. Random Forest model had good precision(0.77), recall(0.53) & f1 score(0.63).
- After model building we had to make sure if our models are over fitted or under fitted or not, so we validated it with cross validation with the help of scikit learn’s model selection package.

Cross validation code & result.



Cross Validations of Model's Accuracy.

```
In [42]: # cross validation for Logistic Regression Model
         cvs = cross_val_score(log_reg, x_scaled, y, cv = 10)
         print('Cross Validation mean score for Logistic Regression model is :', cvs.mean())

         Cross Validation mean score for Logistic Regression model is : 0.8806925853784324

In [43]: # cross validation for Knn Model
         cvs = cross_val_score(knn, x_scaled, y, cv = 10)
         print('Cross Validation mean score for K-Nearest Neighbor model is :', cvs.mean())

         Cross Validation mean score for K-Nearest Neighbor model is : 0.8978162642786043

In [44]: # cross validation for Decision Tree model
         cvs = cross_val_score(DT, x_scaled, y, cv = 10)
         print('Cross Validation ean score for Decision Tree model is :', cvs.mean())

         Cross Validation ean score for Decision Tree model is : 0.8845047589732626

In [45]: # cross validation for Random Forest Model
         cvs = cross_val_score(RF, x_scaled, y, cv = 10)
         print('Cross Validation mean Score for Random Forest Model is :', cvs.mean())

         Cross Validation mean Score for Random Forest Model is : 0.9214763982588327

In [46]: # cross validation for support vector machine model
         cvs = cross_val_score(svc, x_scaled, y, cv = 10)
         print('Cross Validation mean score for Support Vector Classification Model is :', cvs.mean())

         Cross Validation mean score for Support Vector Classification Model is : 0.888612715201143
```

Result of cross validation

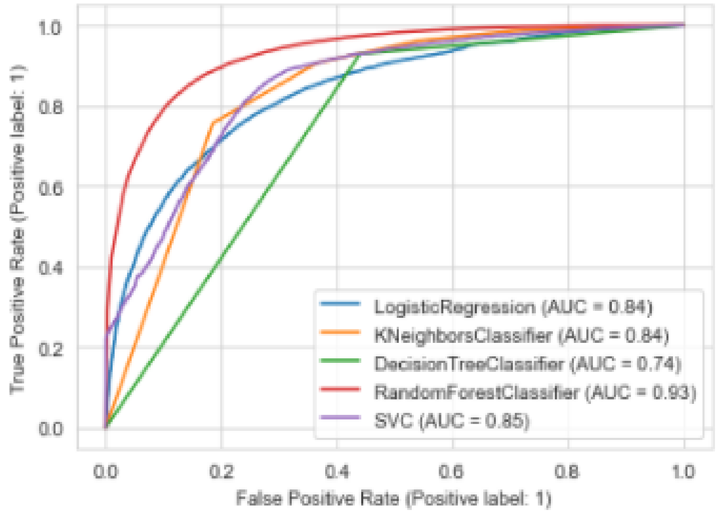
- After cross validation of the model we found that the result of cross validation score are almost same what we had in default models, they had only very few difference as you can see in the table.
- We had decision tree model whose cross validation was even greater than the default model.
- From this step we concluded that our models are neither over fitted nor under fitted.
- But from this we got confused as which model we should choose as best model because as per minimum difference between default accuracy score & cross validation score we had decision tree model. But on the other hand decision tree had least accuracy score among all models & also precision, recall & f1 score was not so good.
- So we decided to evaluate the model with the help of roc_auc plot so that we can choose the model who has highest area under the curve.

Models	Model Accuracy	Cross Validation Score	Difference
Logistic Regression	88.49	88.06	0.43
K-Nearest Neighbors	89.98	89.78	0.2
Decision Tree	88.34	88.45	-0.11
Random Forest	92.28	92.14	0.14
Support Vector Machine	89.29	88.86	0.43

Roc-auc curve plotting code & documentation.

AUC-ROC Curve

```
In [47]: # Plotting roc_auc_curve

         disp = plot_roc_curve(log_reg, x_test, y_test)
         plot_roc_curve(knn, x_test, y_test, ax = disp.ax_)
         plot_roc_curve(DT, x_test, y_test, ax = disp.ax_)
         plot_roc_curve(RF, x_test, y_test, ax = disp.ax_)
         plot_roc_curve(svc, x_test, y_test, ax = disp.ax_)
         plt.show()
```

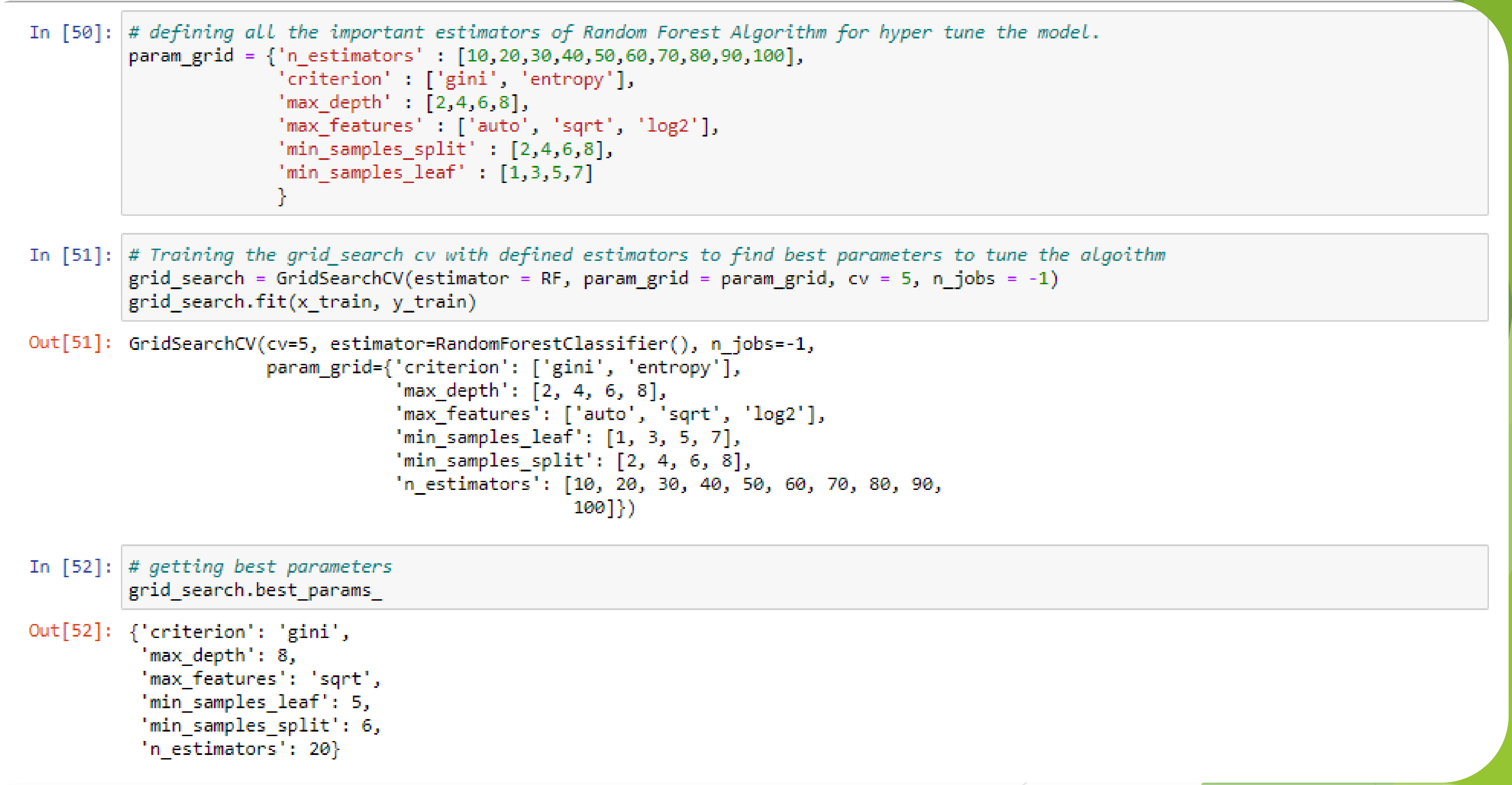


From above ROC plot we can see that Random Forest Model has the highest area under the curve hence we are going to finalize Random Forest as our Micro Cedit Defaulter Prediction Model. Now we will Hypertune the Random Forest Model to see whether we can increase our Model accuracy or not.

Roc - auc plot result & conclusion.

- After plotting roc_auc plot, we were clear about our best model to choose we got the Random Forest model having highest area under the curve.
- Random Forest model was the model which had highest accuracy score, precision, recall & f1 score and this model had even the highest area under the curve.
- Hence we selected Random forest as our project’s model for defaulter prediction & now we wanted to do some hyper parameter tuning in the model so that we could get more accuracy if possible, hence we did the same.

Hyper parameter code & best parameters

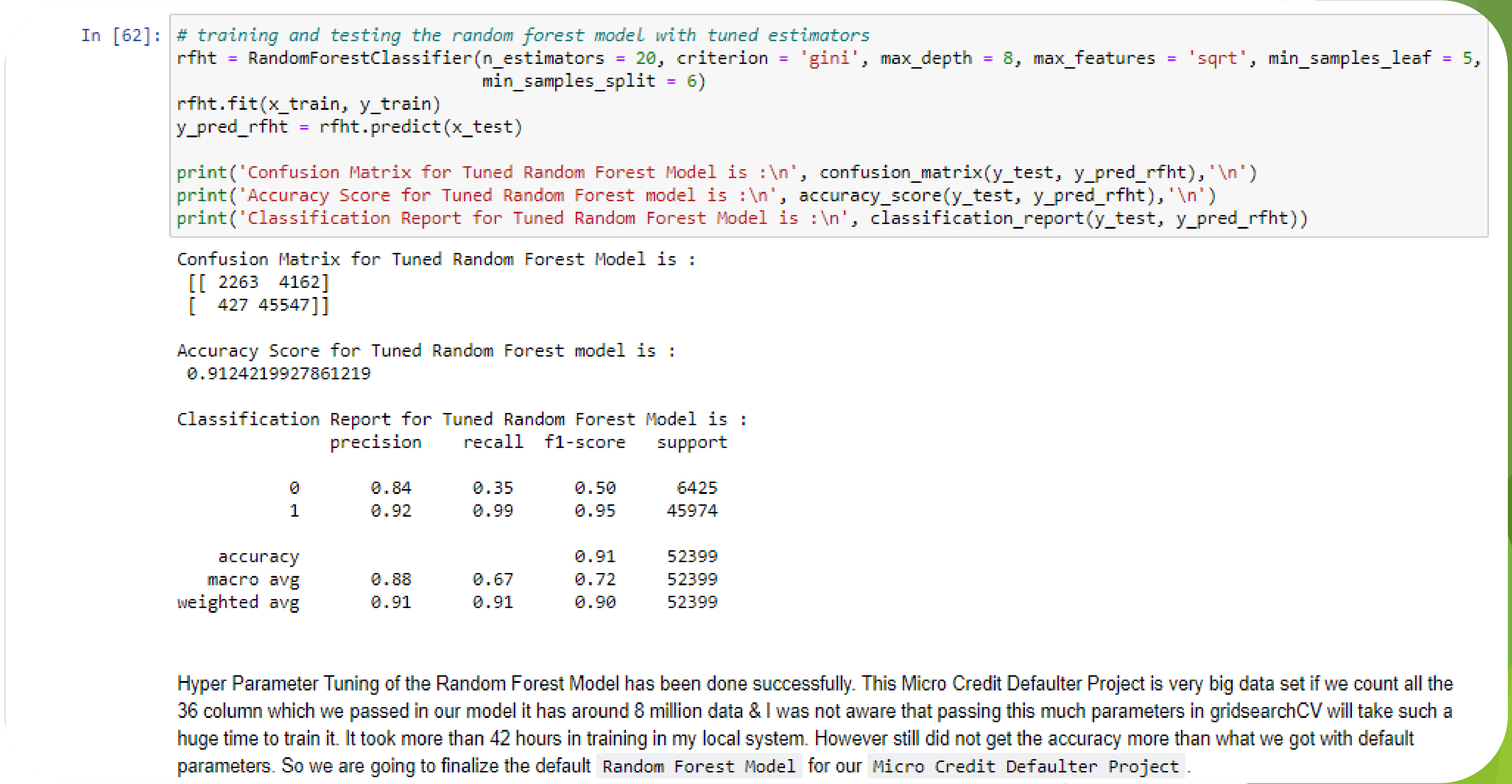


```
In [50]: # defining all the important estimators of Random Forest Algorithm for hyper tune the model.
         param_grid = {'n_estimators' : [10,20,30,40,50,60,70,80,90,100],
                       'criterion' : ['gini', 'entropy'],
                       'max_depth' : [2,4,6,8],
                       'max_features' : ['auto', 'sqrt', 'log2'],
                       'min_samples_split' : [2,4,6,8],
                       'min_samples_leaf' : [1,3,5,7]
                       }

In [51]: # Training the grid_search cv with defined estimators to find best parameters to tune the algoithm
         grid_search = GridSearchCV(estimator = RF, param_grid = param_grid, cv = 5, n_jobs = -1)
         grid_search.fit(x_train, y_train)

Out[51]: GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
                      param_grid={'criterion': ['gini', 'entropy'],
                                  'max_depth': [2, 4, 6, 8],
                                  'max_features': ['auto', 'sqrt', 'log2'],
                                  'min_samples_leaf': [1, 3, 5, 7],
                                  'min_samples_split': [2, 4, 6, 8],
                                  'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90,
                                                   100]})

In [52]: # getting best parameters
         grid_search.best_params_

Out[52]: {'criterion': 'gini',
          'max_depth': 8,
          'max_features': 'sqrt',
          'min_samples_leaf': 5,
          'min_samples_split': 6,
          'n_estimators': 20}
```

Hyper parameter tuning result & conclusion

- Hyper parameter tuning of Random Forest algorithm was the most challenging part of the project.
- The data set was huge and we were aware that if we give large parameters to GridSearchCV for Random Forest it would take a lot of time but we were not aware that it would take more than 48 hours to train.
- We tried to train the GridSearchCV on Google colab there also it failed multiple times, hence we reduced the parameters and tried again.
- Even after giving 48 hours to the hyper tuning the result was not satisfying it gave us score lower than what we had with default parameters.
- Hence proceeded the model saving with default Random Forest model.

Training & testing of Random forest with tuned parameters.

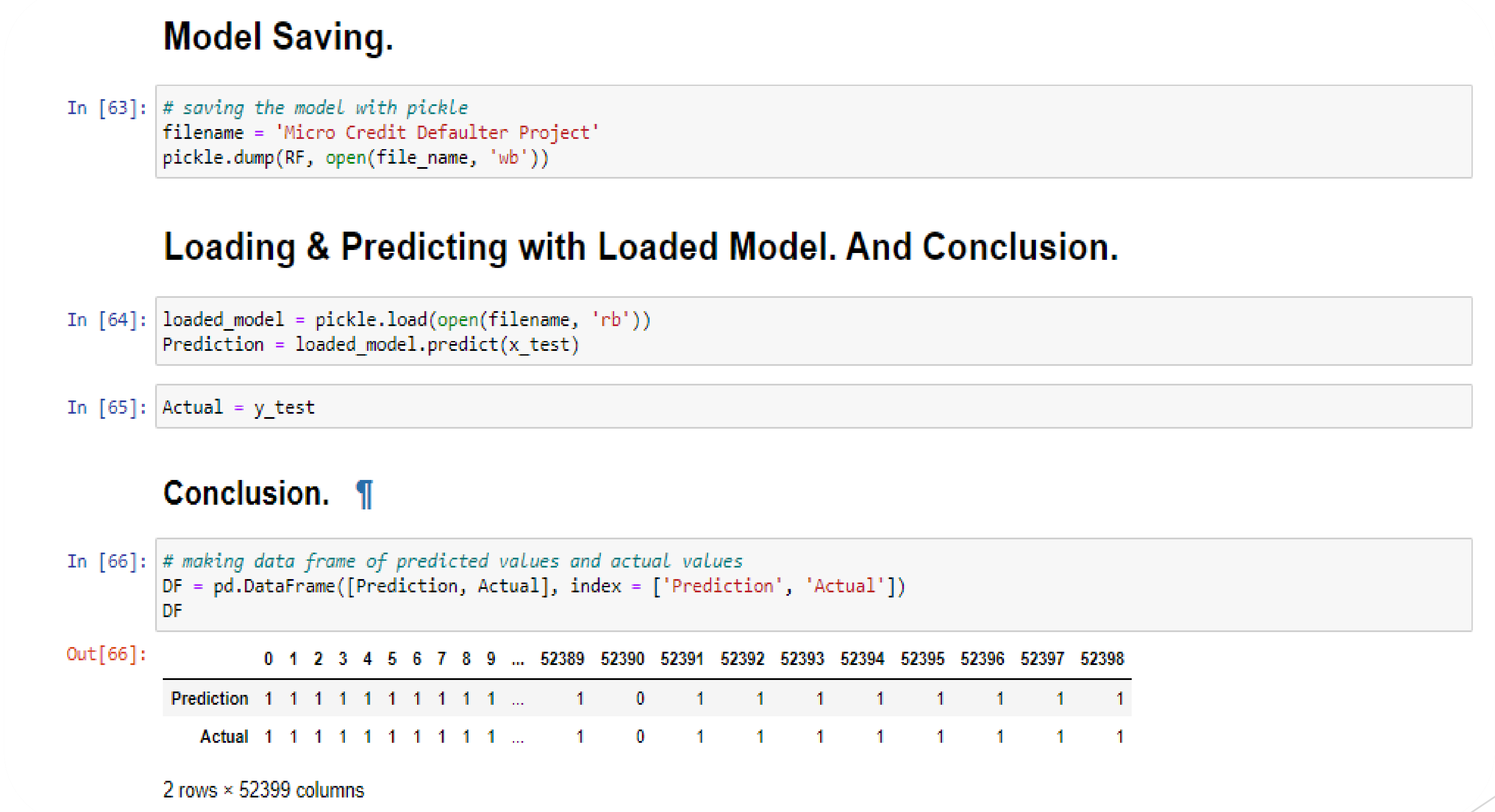


```
In [62]: # training and testing the random forest model with tuned estimators
         rfht = RandomForestClassifier(n_estimators = 20, criterion = 'gini', max_depth = 8, max_features = 'sqrt', min_samples_leaf = 5,
                                       min_samples_split = 6)
         rfht.fit(x_train, y_train)
         y_pred_rfht = rfht.predict(x_test)

         print('Confusion Matrix for Tuned Random Forest Model is :\n', confusion_matrix(y_test, y_pred_rfht),'\n')
         print('Accuracy Score for Tuned Random Forest model is :\n', accuracy_score(y_test, y_pred_rfht),'\n')
         print('Classification Report for Tuned Random Forest Model is :\n', classification_report(y_test, y_pred_rfht))
```

```
Confusion Matrix for Tuned Random Forest Model is :
 [[ 2263  4162]
 [  427 45547]]

Accuracy Score for Tuned Random Forest model is :
 0.9124219927861219

Classification Report for Tuned Random Forest Model is :
               precision    recall  f1-score   support

           0       0.84      0.35      0.50      6425
           1       0.92      0.99      0.95     45974

    accuracy                           0.91     52399
   macro avg       0.88      0.67      0.72     52399
weighted avg       0.91      0.91      0.90     52399
```

Hyper Parameter Tuning of the Random Forest Model has been done successfully. This Micro Credit Defaulter Project is very big data set if we count all the 36 column which we passed in our model it has around 8 million data & I was not aware that passing this much parameters in gridsearchCV will take such a huge time to train it. It took more than 42 hours in training in my local system. However still did not get the accuracy more than what we got with default parameters. So we are going to finalize the default `Random Forest Model` for our `Micro Credit Defaulter Project`.

Final step of project model saving, predicting with loaded model & comparing with actual data



Model Saving.

```
In [63]: # saving the model with pickle
         filename = 'Micro Credit Defaulter Project'
         pickle.dump(RF, open(file_name, 'wb'))
```

Loading & Predicting with Loaded Model. And Conclusion.

```
In [64]: loaded_model = pickle.load(open(filename, 'rb'))
         Prediction = loaded_model.predict(x_test)
```

```
In [65]: Actual = y_test
```

Conclusion. ¶

```
In [66]: # making data frame of predicted values and actual values
         DF = pd.DataFrame([Prediction, Actual], index = ['Prediction', 'Actual'])
         DF
```

```
Out[66]:
            0  1  2  3  4  5  6  7  8  9  ...  52389  52390  52391  52392  52393  52394  52395  52396  52397  52398
Prediction  1  1  1  1  1  1  1  1  1  1  ...      1      0      1      1      1      1      1      1      1      1
    Actual  1  1  1  1  1  1  1  1  1  1  ...      1      0      1      1      1      1      1      1      1      1

2 rows × 52399 columns
```

Model saving & conclusion.

- At the last step of the project we decided to go with the **Random Forest** model with default parameters.
- We saved the model in the local system with the help of **PICKLE**.
- After saving the model we tested it by loading the model & predicting the test data with loaded model.
- After prediction of test data with loaded model we made a Data Frame of actual outcomes and predicted outcomes with loaded model and concluded that.

Thank you!