FLIP ROBO

Micro Credit Defaulter Project

Submitted by:

Raghavulu Patnala

ACKNOWLEDGMENT

Thanks for giving me the opportunity to work in Fliprobo Technologies as Intern and would like to express my gratitude to Data Trained Institute as well for trained me in Data Science Domain.

This helps me to do my projects well and understand the concepts.

Resources used – Google, GitHub, Blogs for conceptual referring.

INTRODUCTION

- ## Business Problem Framing

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

  The client wants some predictions that could help them in further investment and improvement in selection of customers for the credit.

- ## Conceptual Background of the Domain Problem

  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.

  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).

This problem contains data of customers who is defaulter / Non – defaulters and has the main account and data account recharge and total amount of sum amount and its frequency. So, we need to predict for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan.

- Motivation for the Problem Undertaken

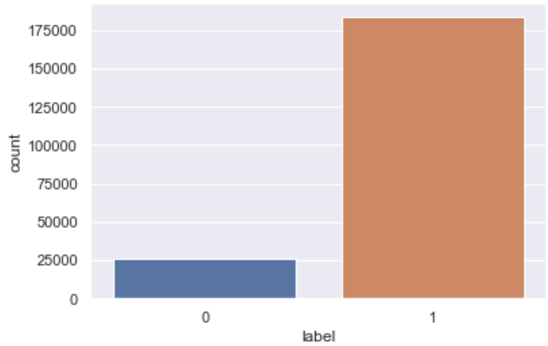
This will help the client to get help on their future investment on telecom industry and that will improve the importance of communication in a person's life, thus, focusing on providing their services and products to low-income families and poor customers that can help them in the need of hour.

Analytical Problem Framing

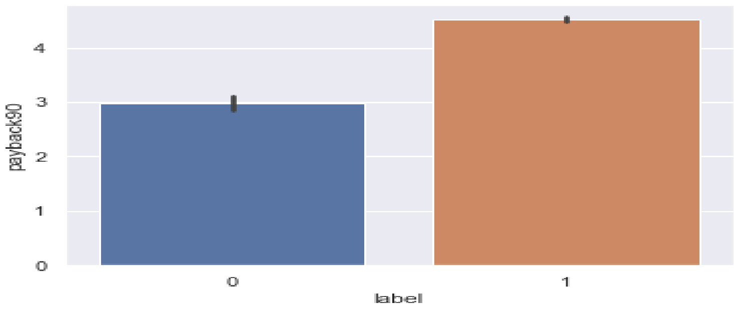
- Data Sources and their formats

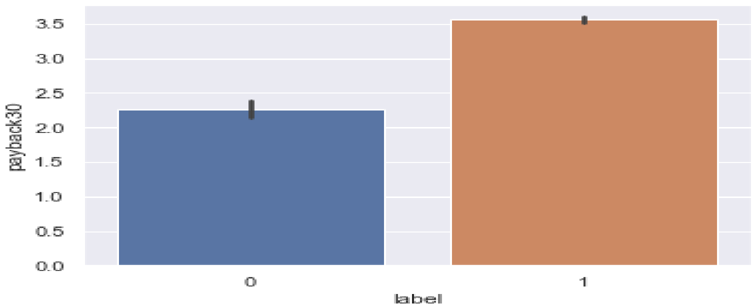
We can see from the below snap that our target variable has more non- defaulters (paying loan on time) than defaulters (not paying loan on time),

```
# We can see that most of the customer will be paying back the loaned amount within 5 days of insurance of loan.
#In this case, Label '1' indicates that the loan has been payed i.e. Non- defaulter, while,
#Label '0' indicates that the loan has not been payed i.e. defaulter.

sns.countplot(d['label'])
plt.show()
```

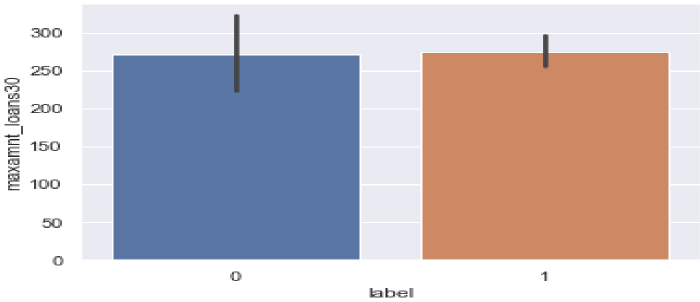


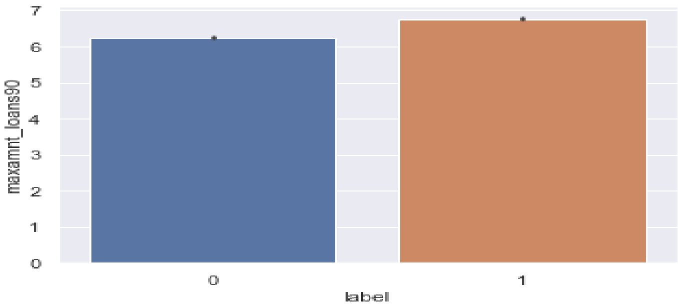
Most of the customers (non- defaulters) are paying back their loan by 3-5 days,



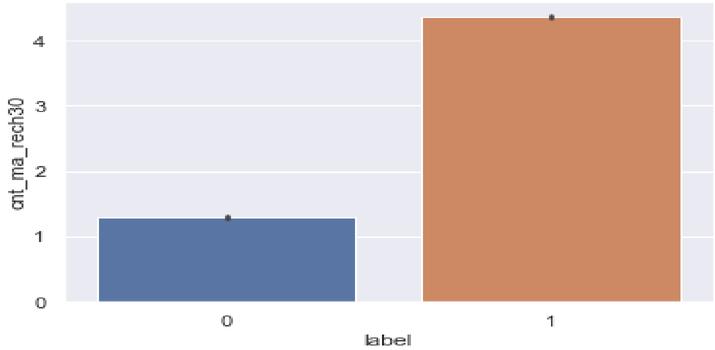


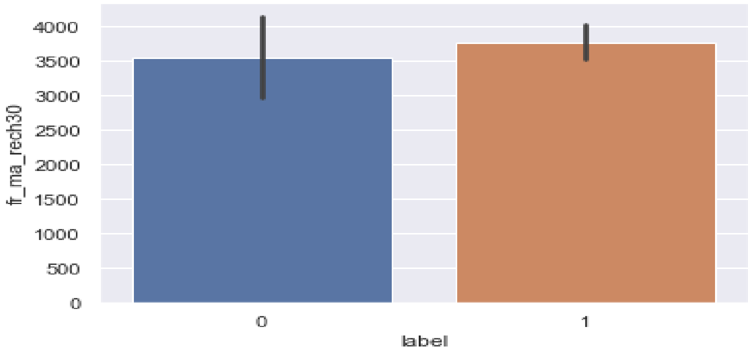
Maximum amount of loans is taken by defaulters in < 30 days and there are 2 options 5Rs and 10Rs which customer needs to payback as 6Rs and 12Rs.



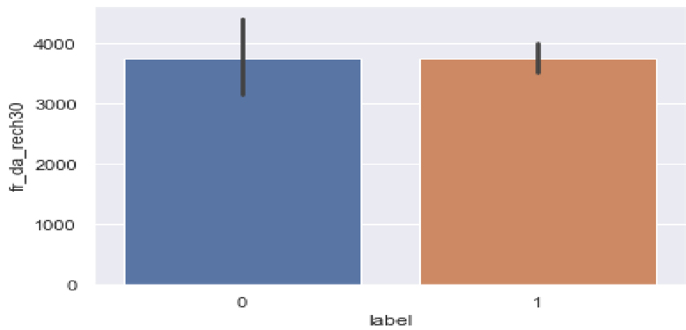


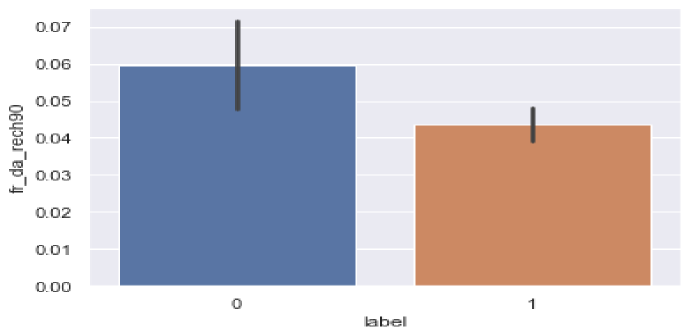
Frequency of main account recharged by a greater number of defaulters in < 30 days.





Frequency of data account recharged by a greater number of defaulters in < 30 days.





- Data Pre-processing Done

  Replacing some of the 0 values to mean, median as it is having 0 values more and customer who got loan has to payback in 30 days and 90 days and frequency of main account and data account recharged and count of data account and main account of recharged.

  If account got recharged and customer needs to payback the loan within their 30 days and 90 days.

  Also, we have outliers as well and Tried applying Z-score method, we ae losing >10% data, So I am removing skewness by using Power transform method.

- Data Inputs- Logic- Output Relationships

  Our target variable is label which indicates the customer is a defaulter who is not paying back or non-defaulter who is paying back the loan properly with some features like how often they are recharging their main and data account and number of times they are recharging, and daily amount spend by customer from main account and average main account balance and number of loans taken by user and maximum amount of loan taken by user in last 30 days or 90 days.

- Hardware and Software Requirements and Tools Used

  1) Pandas is open-source library tool which provides high performance data analysis tool by its powerful data structures. It

helps to shorten the procedure of handling the data with extensive set of features.

2) NumPy is most used package for scientific computing for multi-dimensional array of objects.

3) Other than this, as a pre-processing steps, I Imported Standard scaler for scaling the data.

4) In terms of selecting the which model is best, I Imported Train test split where I am splitting the train data and test data and using cross Val score to calculate whether the model is overfitting or under-fitting and RandomizedsearchCV to check improve the accuracy score.

5) I Imported f1 score, classification report, confusion matrix, roc curve in terms of metrics to calculate the model score.

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

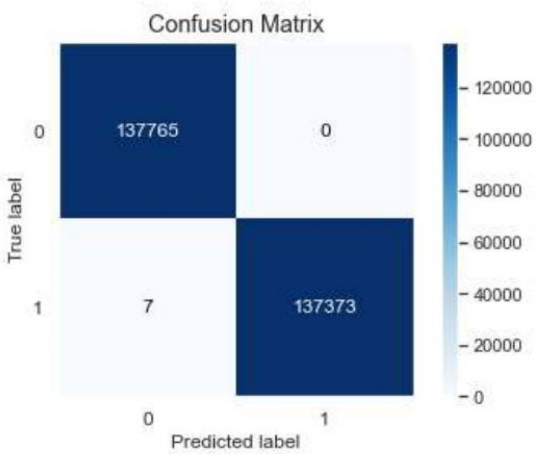
I have used Decision Treen algorithm, Random Forest, Ada Boost and Gradient Boost Algorithm to calculate the score of the model.

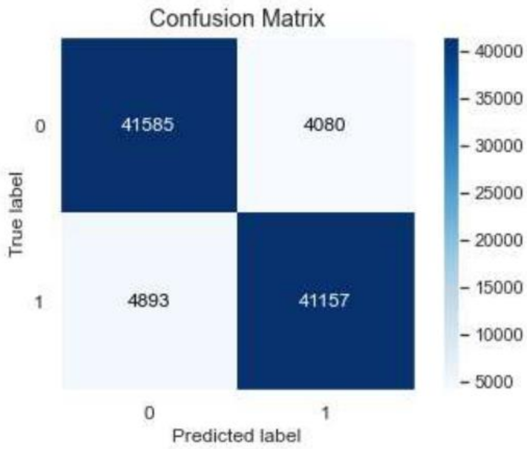
- Run and evaluate selected models

Decision Tree model which has Score – 90 .21% and CV score – 88.23%

```
In [53]: #train result
DecisionTree = DecisionTreeClassifier()
DecisionTree.fit(x_train, y_train)
y_pred =DecisionTree .predict(x_train)
accuracy = classification_report(y_train, y_pred)
print(accuracy)
print(skplt.metrics.plot_confusion_matrix(y_train, y_pred))

#test result
DecisionTree = DecisionTreeClassifier()
DecisionTree.fit(x_train, y_train)
y_pred =DecisionTree .predict(x_test)
accuracy = classification_report(y_test, y_pred)
print(accuracy)
print(skplt.metrics.plot_confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    137765
           1       1.00      1.00      1.00    137380

    accuracy                           1.00    275145
   macro avg       1.00      1.00      1.00    275145
weighted avg       1.00      1.00      1.00    275145

AxesSubplot(0.125,0.125;0.62x0.755)
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     45665
           1       0.91      0.89      0.90     46050

    accuracy                           0.90     91715
   macro avg       0.90      0.90      0.90     91715
weighted avg       0.90      0.90      0.90     91715

AxesSubplot(0.125,0.125;0.62x0.755)
```





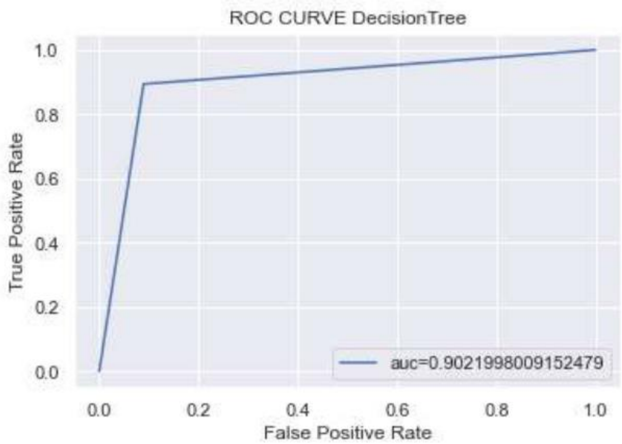

```
In [54]: print("Training accuracy::",DecisionTree.score(x_train,y_train))
         print("Test accuracy::",DecisionTree.score(x_test,y_test))
```

```
Training accuracy:: 0.99997455886896
Test accuracy:: 0.9021643133620455
```

```
In [55]: print(cross_val_score(DecisionTree,x,y,cv=5).mean())
```

```
0.8823428273668334
```

```
In [56]: #roc_curve plot to check the socre of Decisiontree
         fpr, tpr, _= roc_curve(y_test, y_pred)
         auc_score = roc_auc_score(y_test, y_pred)
         plt.plot(fpr, tpr, label="auc="+str(auc_score))
         plt.box(True)
         plt.title('ROC CURVE DecisionTree')
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.legend(loc=4)
         plt.grid(True)
         plt.show()
         print('The Score for the ROC Curve is : {}%'.format(round(auc_score,4)*100))
```

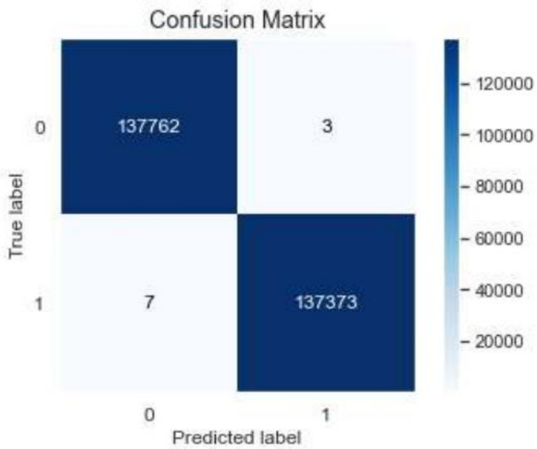


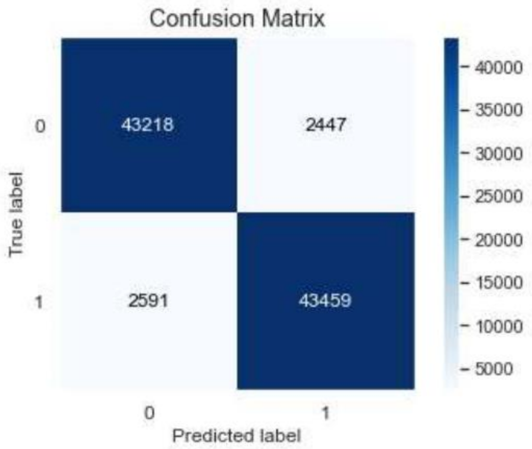

```
The Score for the ROC Curve is : 90.22%
```

Random Forest model has score – 94.5% and CV score – 91.2%

In [57]:
```
#train result
RFC = RandomForestClassifier()
RFC.fit(x_train, y_train)
y_pred =RFC .predict(x_train)
accuracy = classification_report(y_train, y_pred)
print(accuracy)
print(skplt.metrics.plot_confusion_matrix(y_train, y_pred))


#test result
RFC = RandomForestClassifier()
RFC.fit(x_train, y_train)
y_pred =RFC .predict(x_test)
accuracy = classification_report(y_test, y_pred)
print(accuracy)
print(skplt.metrics.plot_confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    137765
           1       1.00      1.00      1.00    137380

    accuracy                           1.00    275145
   macro avg       1.00      1.00      1.00    275145
weighted avg       1.00      1.00      1.00    275145

AxesSubplot(0.125,0.125;0.62x0.755)
              precision    recall  f1-score   support

           0       0.94      0.95      0.94     45665
           1       0.95      0.94      0.95     46050

    accuracy                           0.95     91715
   macro avg       0.95      0.95      0.95     91715
weighted avg       0.95      0.95      0.95     91715

AxesSubplot(0.125,0.125;0.62x0.755)
```





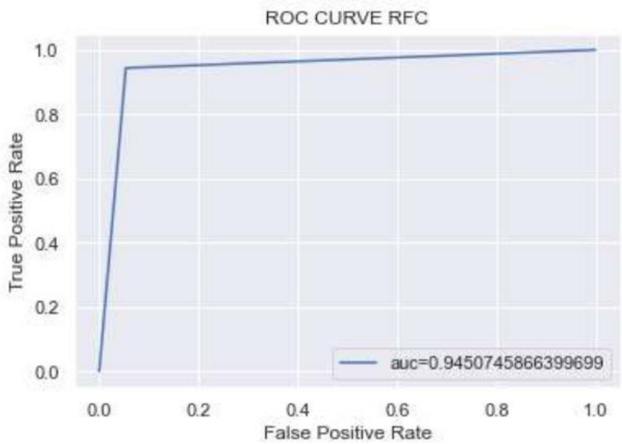
In [58]:
```
print("Training accuracy::",RFC.score(x_train,y_train))
print("Test accuracy::",RFC.score(x_test,y_test))
```

```
Training accuracy:: 0.9999600210797943
Test accuracy:: 0.9450689636373548
```

In [59]:
```
print(cross_val_score(RFC,x,y,cv=5).mean())
```

```
0.9197536192026773
```

In [60]:
```
#roc_curve plot to check the socre of RFC
fpr, tpr, _= roc_curve(y_test, y_pred)
auc_score = roc_auc_score(y_test, y_pred)
plt.plot(fpr, tpr, label="auc="+str(auc_score))
plt.box(True)
plt.title('ROC CURVE RFC')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.grid(True)
plt.show()
print('The Score for the ROC Curve is : {}%'.format(round(auc_score,4)*100))
```

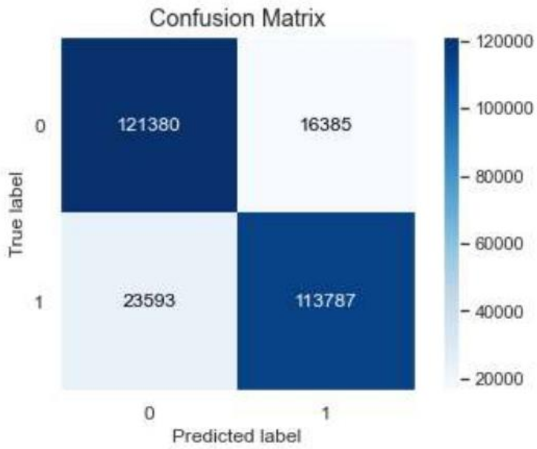


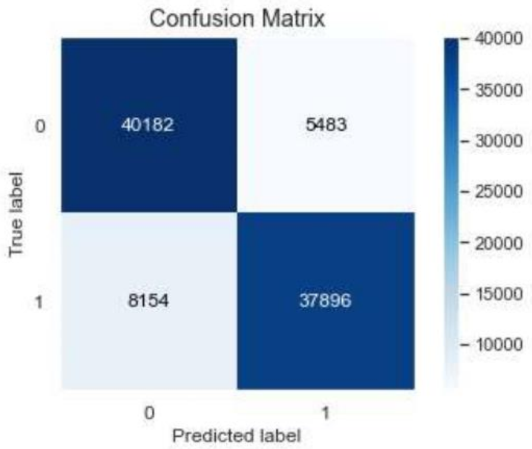

```
The Score for the ROC Curve is : 94.51%
```

Ada Boost has score – 85.1% and CV Score – 90.93%

In [61]:
```
#train result
adb = AdaBoostClassifier()
adb.fit(x_train, y_train)
y_pred =adb .predict(x_train)
accuracy = classification_report(y_train, y_pred)
print(accuracy)
print(skplt.metrics.plot_confusion_matrix(y_train, y_pred))


#test result
adb = AdaBoostClassifier()
adb.fit(x_train, y_train)
y_pred =adb .predict(x_test)
accuracy = classification_report(y_test, y_pred)
print(accuracy)
print(skplt.metrics.plot_confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.88      0.86    137765
           1       0.87      0.83      0.85    137380

    accuracy                           0.85    275145
   macro avg       0.86      0.85      0.85    275145
weighted avg       0.86      0.85      0.85    275145

AxesSubplot(0.125,0.125;0.62x0.755)
              precision    recall  f1-score   support

           0       0.83      0.88      0.85     45665
           1       0.87      0.82      0.85     46050

    accuracy                           0.85     91715
   macro avg       0.85      0.85      0.85     91715
weighted avg       0.85      0.85      0.85     91715

AxesSubplot(0.125,0.125;0.62x0.755)
```





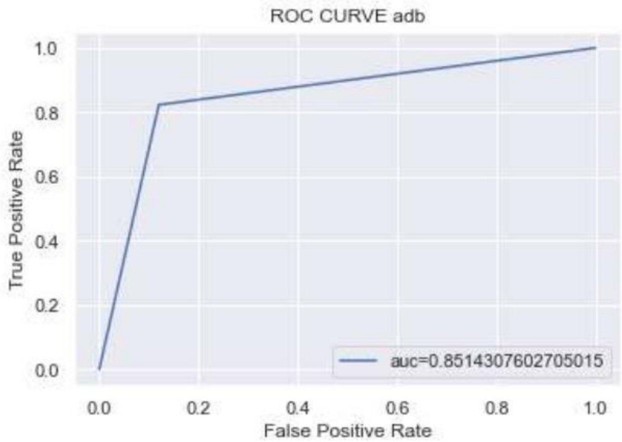

```
In [62]: print("Training accuracy::",adb.score(x_train,y_train))
         print("Test accuracy::",adb.score(x_test,y_test))
```

```
Training accuracy:: 0.8547020661832851
Test accuracy:: 0.8513111268603827
```

```
In [65]: print(cross_val_score(adb,x,y,cv=10).mean())
```

```
0.909376306849906
```

```
In [66]: #roc_curve plot to check the socre of AdaBoostClassifier
         fpr, tpr, _= roc_curve(y_test, y_pred)
         auc_score = roc_auc_score(y_test, y_pred)
         plt.plot(fpr, tpr, label="auc="+str(auc_score))
         plt.box(True)
         plt.title('ROC CURVE adb')
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.legend(loc=4)
         plt.grid(True)
         plt.show()
         print('The Score for the ROC Curve is : {}%'.format(round(auc_score,4)*100))
```

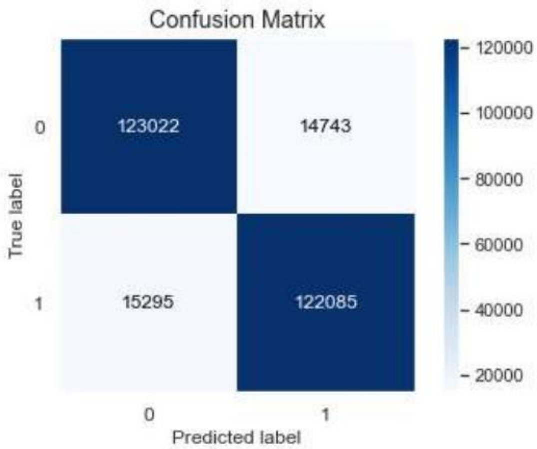


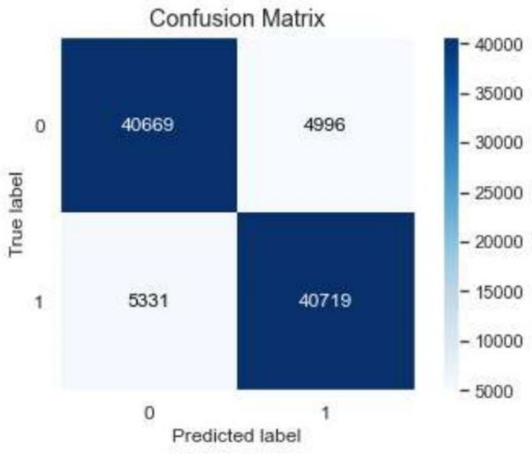

```
The Score for the ROC Curve is : 85.14%
```

Gradient Boost model has score – 88.7% and CV Score – 91.7%

In [67]:
```
#train result
gbc = GradientBoostingClassifier()
gbc.fit(x_train, y_train)
y_pred =gbc .predict(x_train)
accuracy = classification_report(y_train, y_pred)
print(accuracy)
print( skplt.metrics.plot_confusion_matrix(y_train, y_pred))


#test result
gbc = GradientBoostingClassifier()
gbc.fit(x_train, y_train)
y_pred =gbc .predict(x_test)
accuracy = classification_report(y_test, y_pred)
print(accuracy)
print( skplt.metrics.plot_confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.89      0.89    137765
           1       0.89      0.89      0.89    137380

    accuracy                           0.89    275145
   macro avg       0.89      0.89      0.89    275145
weighted avg       0.89      0.89      0.89    275145

AxesSubplot(0.125,0.125;0.62x0.755)
              precision    recall  f1-score   support

           0       0.88      0.89      0.89     45665
           1       0.89      0.88      0.89     46050

    accuracy                           0.89     91715
   macro avg       0.89      0.89      0.89     91715
weighted avg       0.89      0.89      0.89     91715

AxesSubplot(0.125,0.125;0.62x0.755)
```





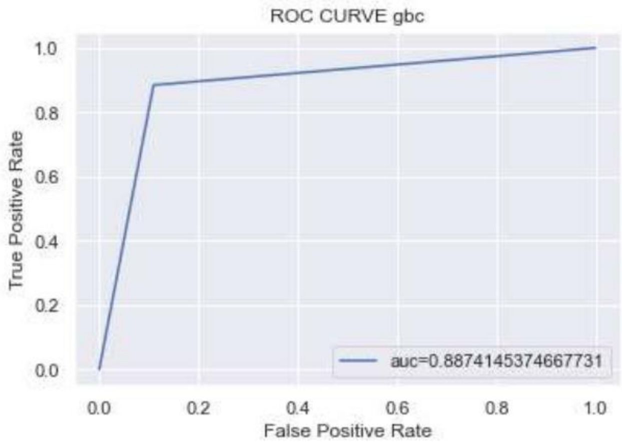
In [68]:
```
print("Training accuracy::",gbc.score(x_train,y_train))
print("Test accuracy::",gbc.score(x_test,y_test))
```

```
Training accuracy:: 0.890828472260081
Test accuracy:: 0.8874011884642643
```

In [72]:
```
print(cross_val_score(gbc,x,y,cv=5).mean())
```

```
0.9179787392483606
```

In [70]:
```
#roc_curve plot to check the socre of GradientBoostClassifier
fpr, tpr, _= roc_curve(y_test, y_pred)
auc_score = roc_auc_score(y_test, y_pred)
plt.plot(fpr, tpr, label="auc="+str(auc_score))
plt.box(True)
plt.title('ROC CURVE gbc')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.grid(True)
plt.show()
print('The Score for the ROC Curve is : {}%'.format(round(auc_score,4)*100))
```




```
The Score for the ROC Curve is : 88.74%
```

Random forest is the model which is having high accuracy score among all other models but when comparing

- Key Metrics for success in solving problem under consideration

  Used F1 Score for calculating the accuracy score as the target variables classes are im-balanced and accuracy score metric won't give correct results as it may take classes with more count.

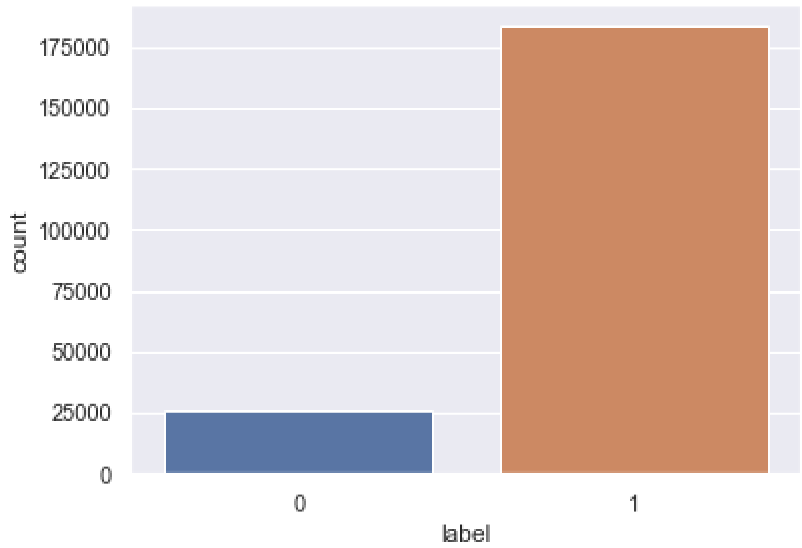
  Classification report will display the overview of accuracy, precision, recall, f1-score, support and weighted average.

  Confusion Matrix for calculating true positive and true negative.

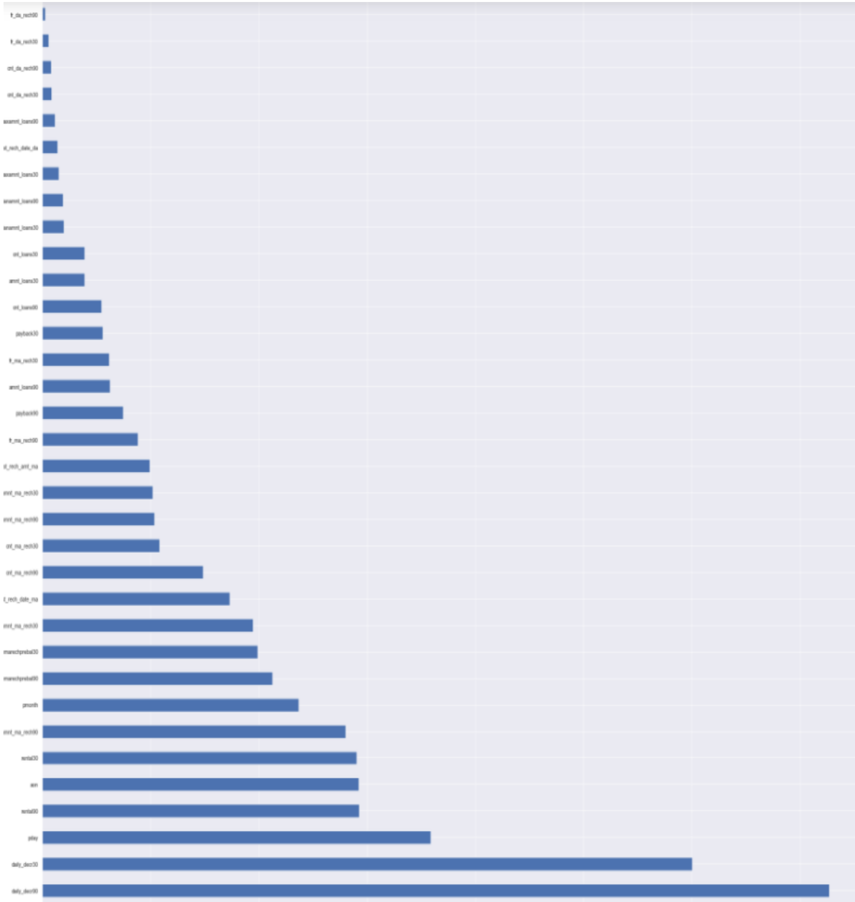
- Visualizations

  Target variable plot where it shows the classes are im-balanced.

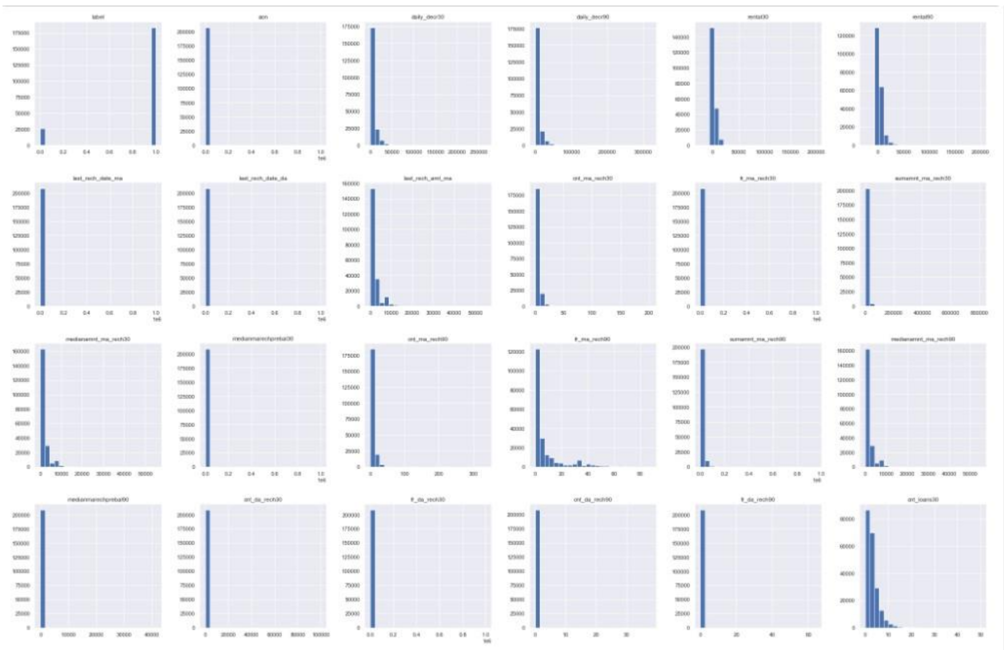
```
# Checking target varibale is im-balanced or not,

sns.countplot(Y)
plt.show()
```

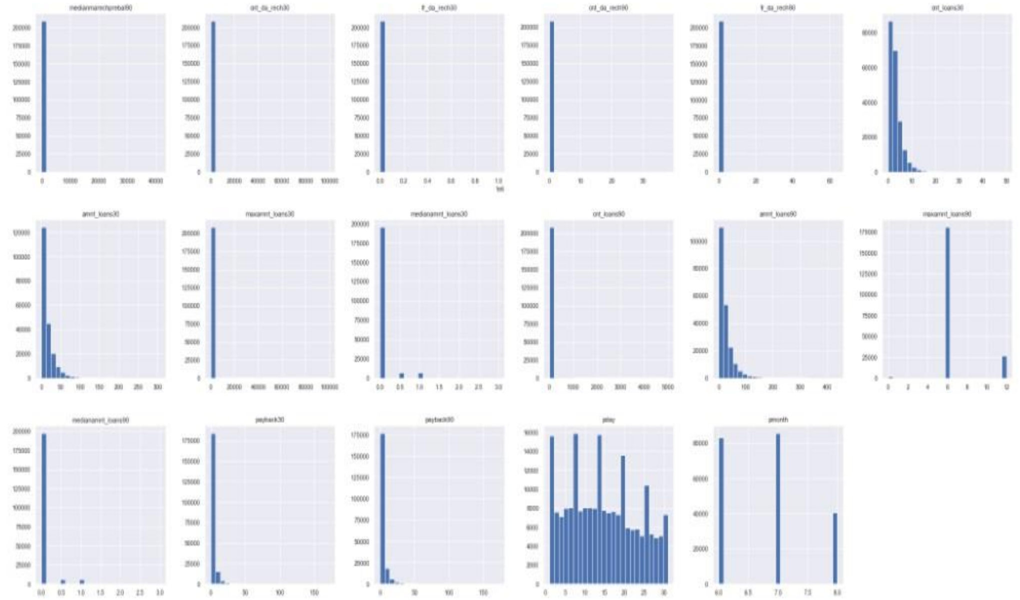


Feature Importance,



Histogram Plots of all columns,



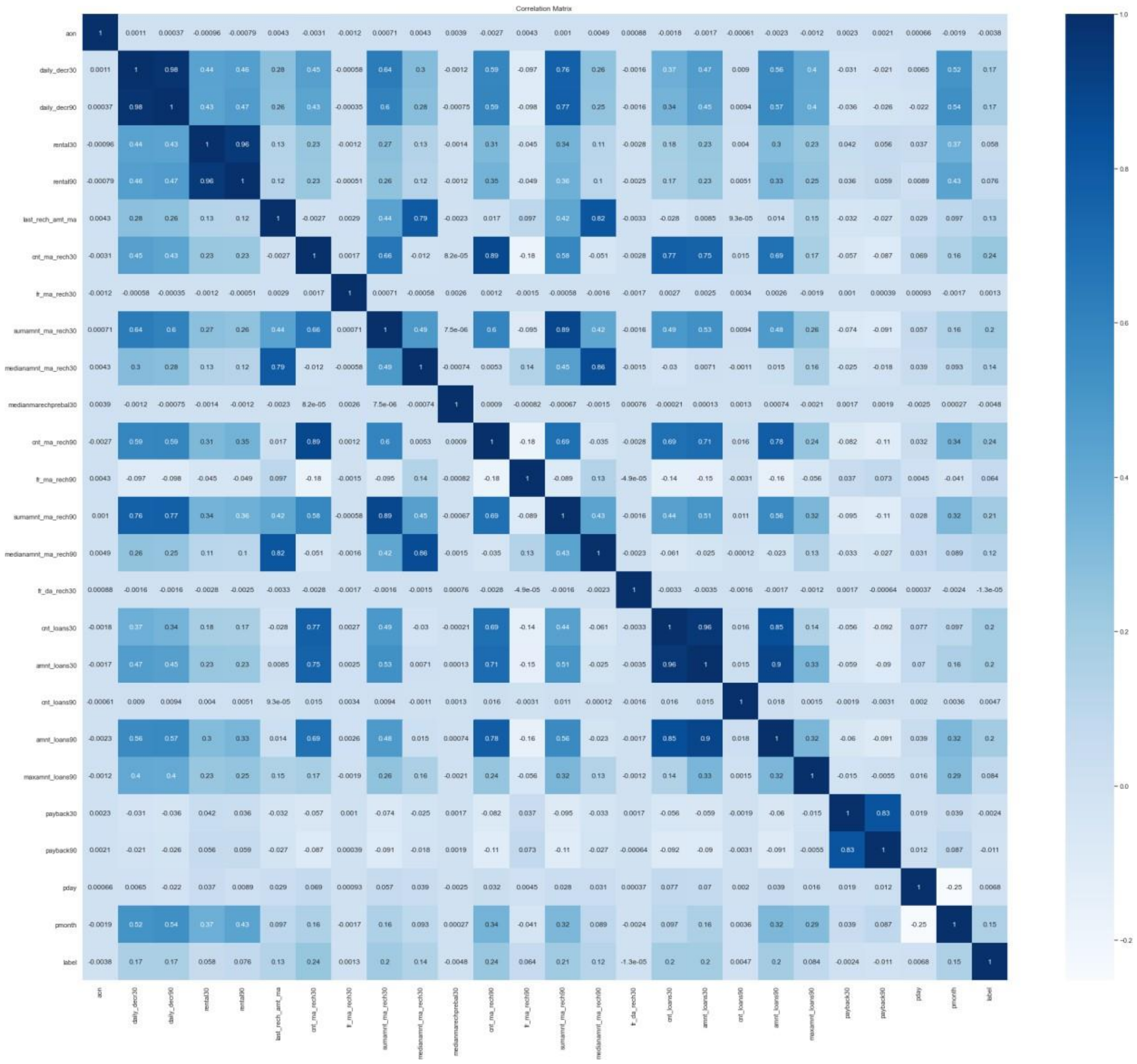


medianmarechprebal90
cnt_da_rech30
fr_da_rech30
cnt_da_rech90
fr_da_rech90
cnt_loans30
amnt_loans30
maxamnt_loans30
medianamnt_loans30
cnt_loans90
amnt_loans90
maxamnt_loans90
medianamnt_loans90
payback30
payback90
pday
pmonth

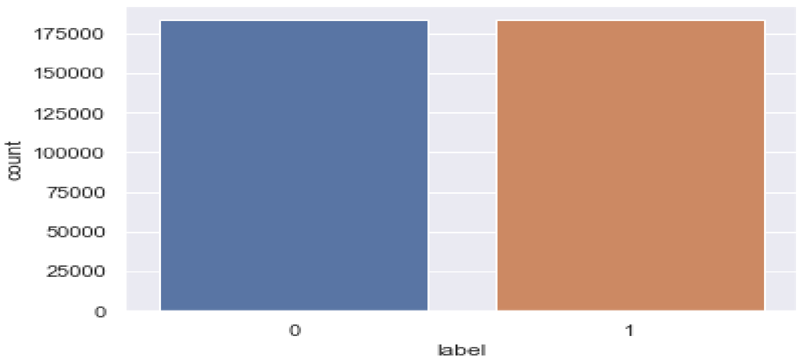
Box plot-

aon
daily_decr30
daily_decr90
rental30
rental90
last_rech_date_ma
last_rech_date_da
last_rech_amt_ma
cnt_ma_rech30
fr_ma_rech30
sumamnt_ma_rech30
medianamnt_ma_rech30
medianmarechprebal30
cnt_ma_rech90
fr_ma_rech90
sumamnt_ma_rech90
medianamnt_ma_rech90
medianmarechprebal90
cnt_da_rech30
fr_da_rech30
cnt_da_rech90
fr_da_rech90
cnt_loans30
amnt_loans30
maxamnt_loans30
medianamnt_loans30
cnt_loans90
amnt_loans90
maxamnt_loans90
medianamnt_loans90

- Interpretation of the Results
  Correlation matrix after dropping less importance features and high skewed data,



After using SMOTE () technique for balancing the im-balanced class,



Handling skew-

In [35]:
```
x.apply(np.sqrt)
```

Out[35]:

	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last
0	16.492423	55.272507	55.363797	14.836779	16.128546	1.414214	60.927995	
1	26.683328	110.099955	110.112443	60.755740	60.755740	4.472136	60.927995	
2	23.130067	37.389838	37.389838	30.002167	30.002167	1.732051	60.927995	
3	15.524175	4.607385	4.607385	12.626163	12.626163	6.403124	60.927995	
4	30.773365	12.272707	12.272707	33.149661	33.149661	2.000000	60.927995	
...	...	...	...	...	...	...	...	
209588	20.099751	12.323649	12.323649	33.002879	33.002879	1.000000	60.927995	
209589	32.787193	6.077499	6.077499	41.573549	41.573549	2.000000	60.927995	
209590	31.827661	108.826062	109.107058	76.562589	94.303765	1.732051	60.927995	
209591	41.617304	111.750742	112.135498	20.293595	31.378018	1.414214	6.164414	
209592	39.761791	67.002701	67.341072	21.998182	25.123694	3.605551	60.927995	

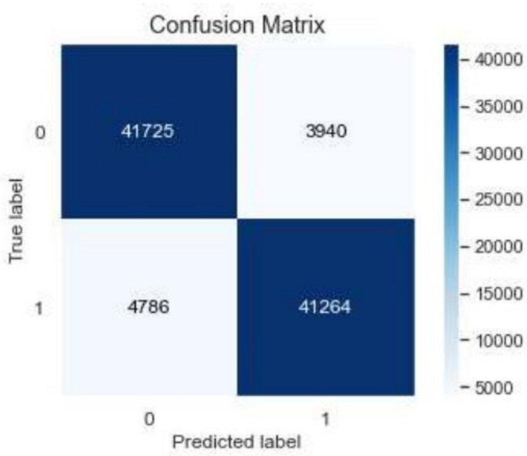
209592 rows × 34 columns

In [36]:
```
# applying power transform method for removing skewness
# Tried Zscore method and data loss % is more.

from sklearn.preprocessing import power_transform
df1 = power_transform(x, method ='yeo-johnson')

df1= pd.DataFrame(df1,columns=x.columns)
```

Random forest is the model which is having high accuracy score among all other models but when comparing with cross val score , the model which is having less difference between score and cv score is best model and here we can tell " decision tree " as best model.

```
In [80]: #final model accuracy,

model = DecisionTreeClassifier(min_samples_split = 18, min_samples_leaf = 4,
                               max_features = 12, criterion = 'entropy',splitter = 'best'

model.fit(x_train,y_train)
y_pred = model.predict(x_test)


print("F1 score \n", f1_score(y_test,y_pred))
print("-------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("-------------------------------------------------------\n")
print("Confusion Matrix \n", skplt.metrics.plot_confusion_matrix(y_test, y_pred))
print("ROC AUC Score \n", roc_auc_score(y_test,y_pred))
```

```
F1 score
 0.9043767944418875
-------------------------------------------------------

Classification Report
               precision    recall  f1-score   support

           0       0.90      0.91      0.91     45665
           1       0.91      0.90      0.90     46050

    accuracy                           0.90     91715
   macro avg       0.90      0.90      0.90     91715
weighted avg       0.91      0.90      0.90     91715

-------------------------------------------------------

Confusion Matrix
 AxesSubplot(0.125,0.125;0.62x0.755)
ROC AUC Score
 0.9048944842491101
```

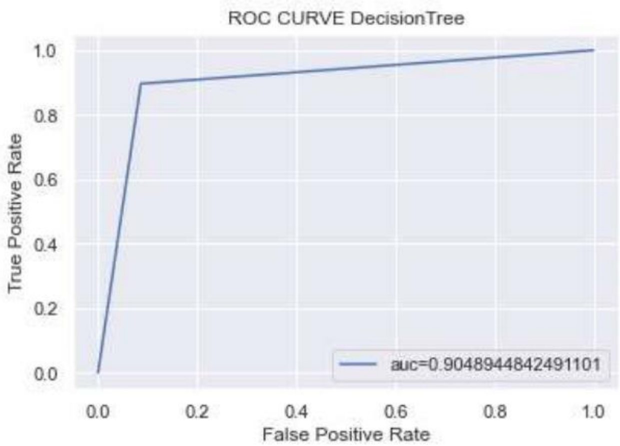


Model has improved the accuracy from 89% to 91%

Final model accuracy Decision tree score – 91%

Roc curve of final model,

In [82]:
```
#Roc Curve for final model,
fpr, tpr, _= roc_curve(y_test, y_pred)
auc_score = roc_auc_score(y_test, y_pred)
plt.plot(fpr, tpr, label="auc="+str(auc_score))
plt.box(True)
plt.title('ROC CURVE DecisionTree')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.grid(True)
plt.show()
print('The Score for the ROC Curve is : {}%'.format(round(auc_score,4)*100))
```




```
The Score for the ROC Curve is : 90.49000000000001%
```

Saving Model

In [85]:
```
# Saving the model,

import joblib
joblib.dump(model,'Micro_Credit_Defaulter_Predection')
```

Out[85]:
```
['Micro_Credit_Defaulter_Predection']
```

CONCLUSION

- Key Findings and Conclusions of the Study, Limitations

  We can tell that target variable is im-balanced and need to balance that and data loss is more actually and need to handle that as well as we can’t lose >8% of data.

  Dealing with huge dataset has taken lot of time for running each algorithm and hyper parameter has taken more time to train the data and it was a nice experience that I have learnt so many things by worked on this project.