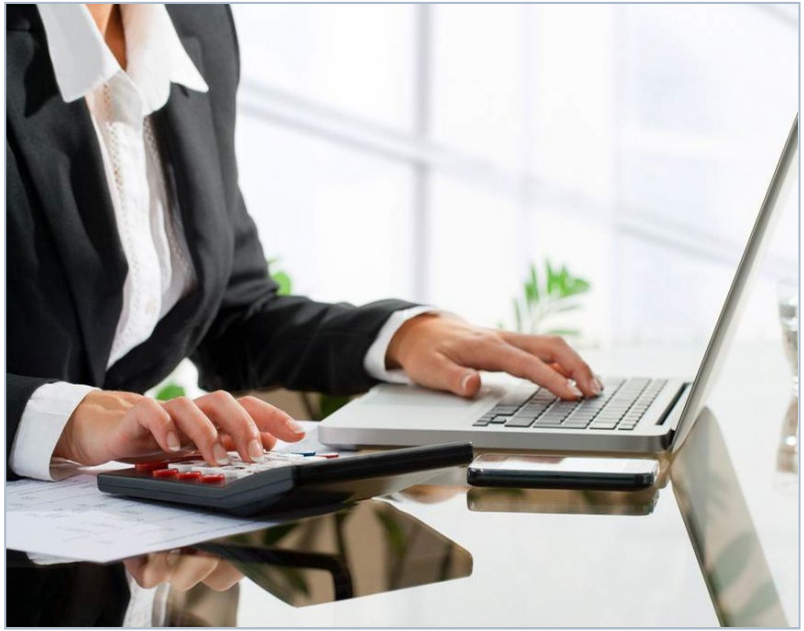
Micro credit defaulter

Submitted by:pailla kusa raju

Contents

- Overview.
- Problem Statement.
- Micro Credit
- Exploratory data analysis.
- Visualizations.
- Analysis.
- Data cleaning
- Model Selection
- Hyper Parameter Tunning.
- ROC-AUC Curve.
- Saving the model and predictions from saved best model.
- Conclusion.

Overview:



✓In this particular presentation we will be looking on:

- How to analyze the dataset of Micro Credit Defaulters.
- What are the EDA steps in cleaning the dataset.
- Overall analysis on the problem.
- Model building from the cleaned dataset.
- Predicting defaulters for saved model.

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.

Micro Credit:

Add your title

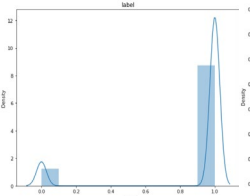
- Microcredit is an **extremely small loan given to those who lack a steady source of income**, collateral. It is used as a way to obtain a loan, acting as a protection against potential loss for the lender should the borrower default in his payments., or any credit history.

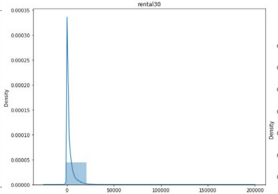
Exploratory Data Analysis:

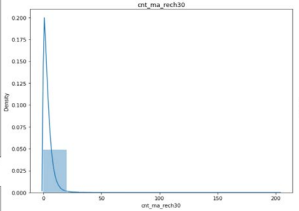
- ✓ First, I have imported all required libraries and I have imported the dataset which was in csv format.
- ✓ Then I did all the statistical analysis like checking shape, nunique, value counts, info, describe etc…..
- ✓ While checking the value counts of the datasets I found some columns with more than 90% zero values and some with more than 70% zero values, so these columns will create skewness in datasets so I decided to drop those columns.
- ✓ When i Checked for the null values I found there is no null values in the dataset.
- ✓ Then I have extracted day, month, year from pdate.

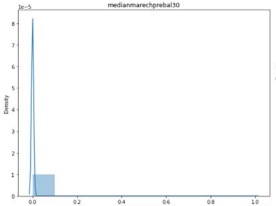
Visualization:

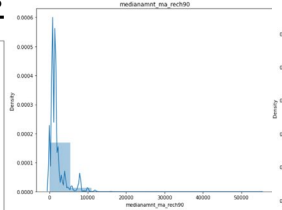
Univariate Analysis:

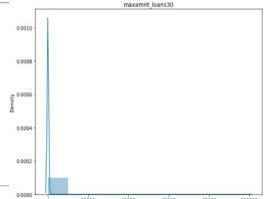


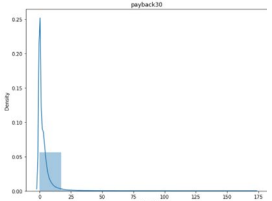


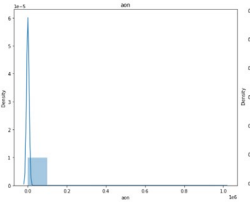


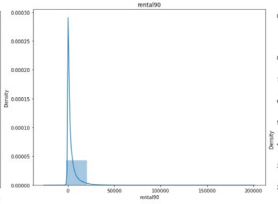


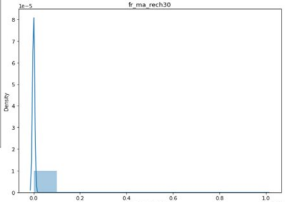


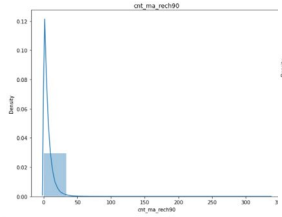


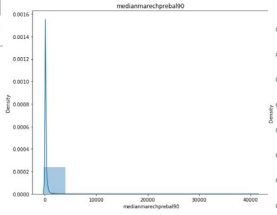


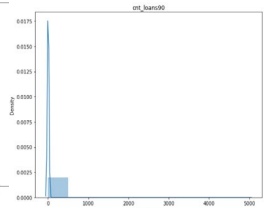


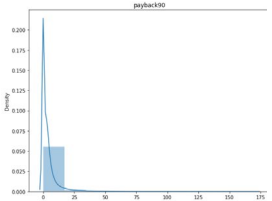


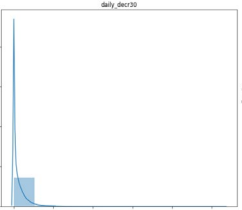


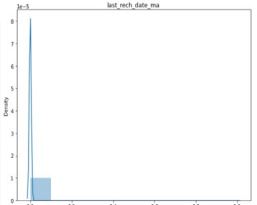


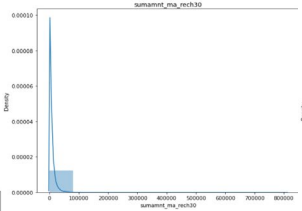


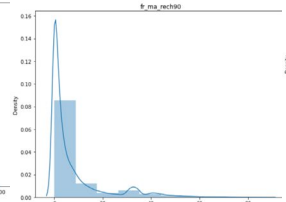


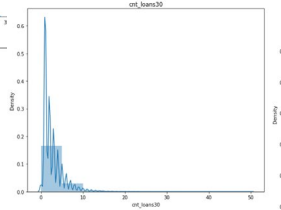


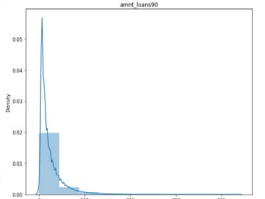


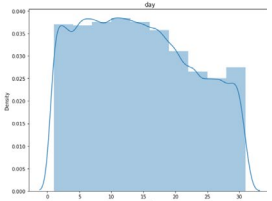


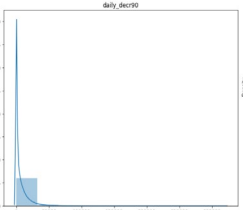


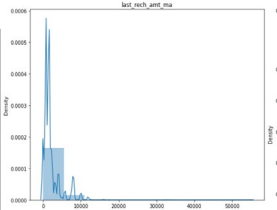


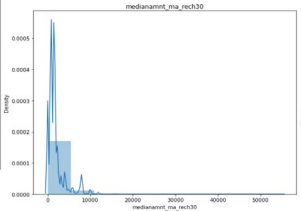


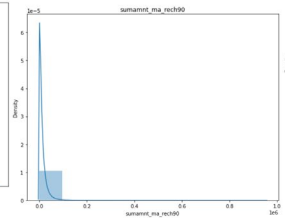


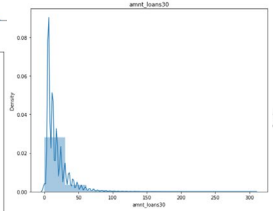


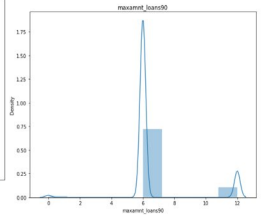


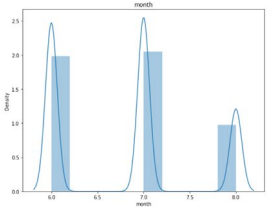




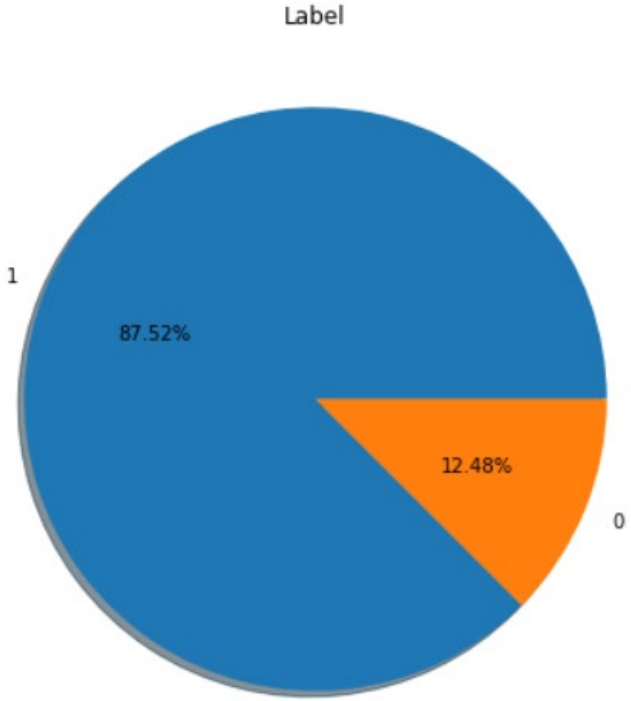




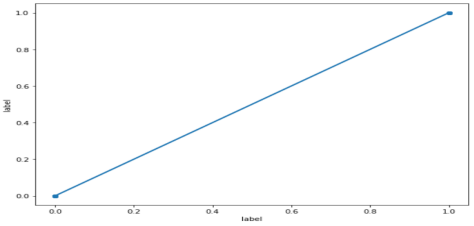


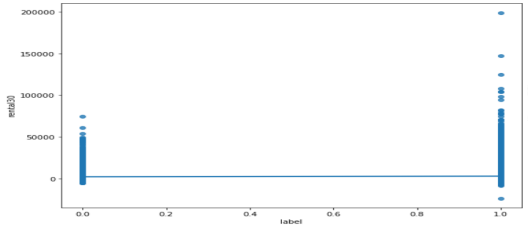


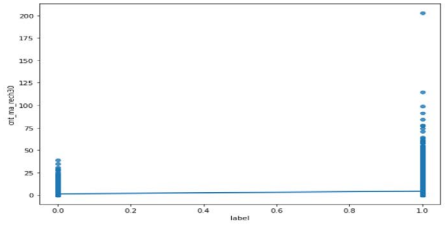
Univariate Target Column:

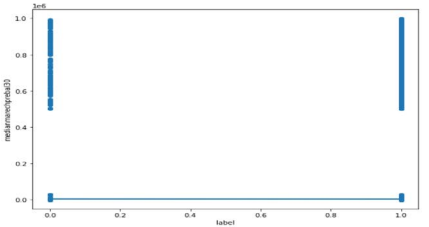


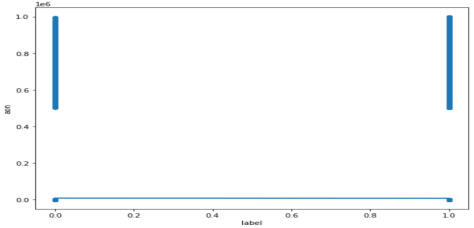
Bivariate Analysis:

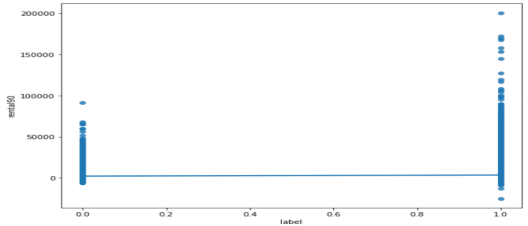


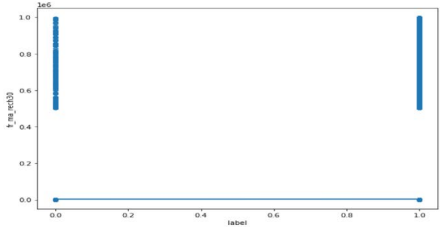


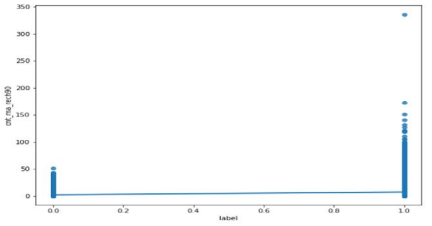


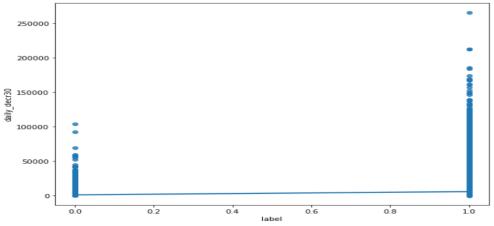


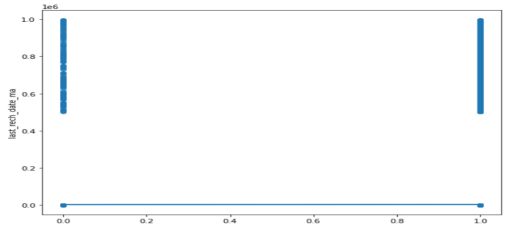


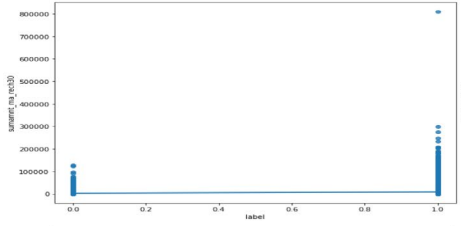


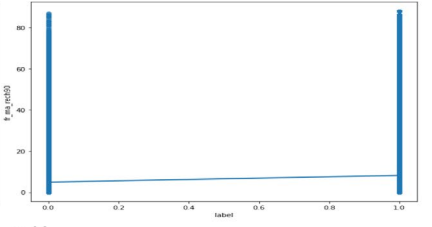


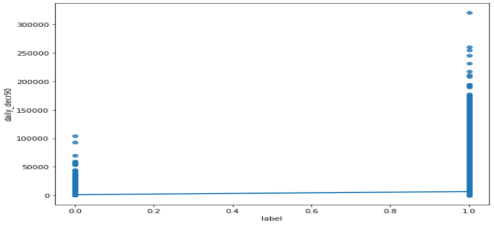


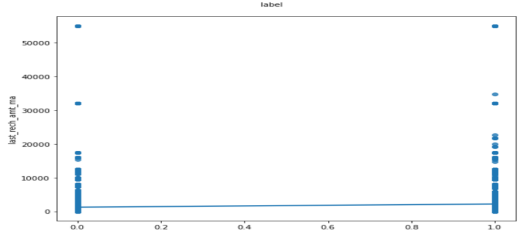


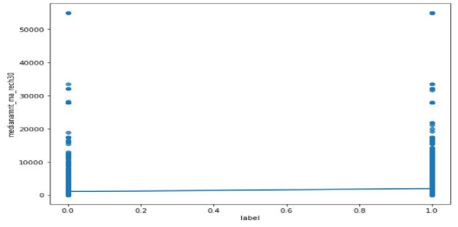


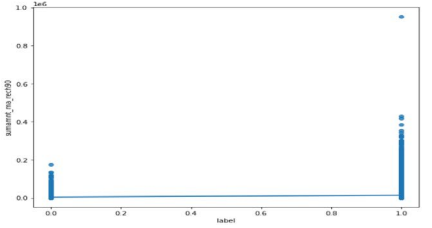




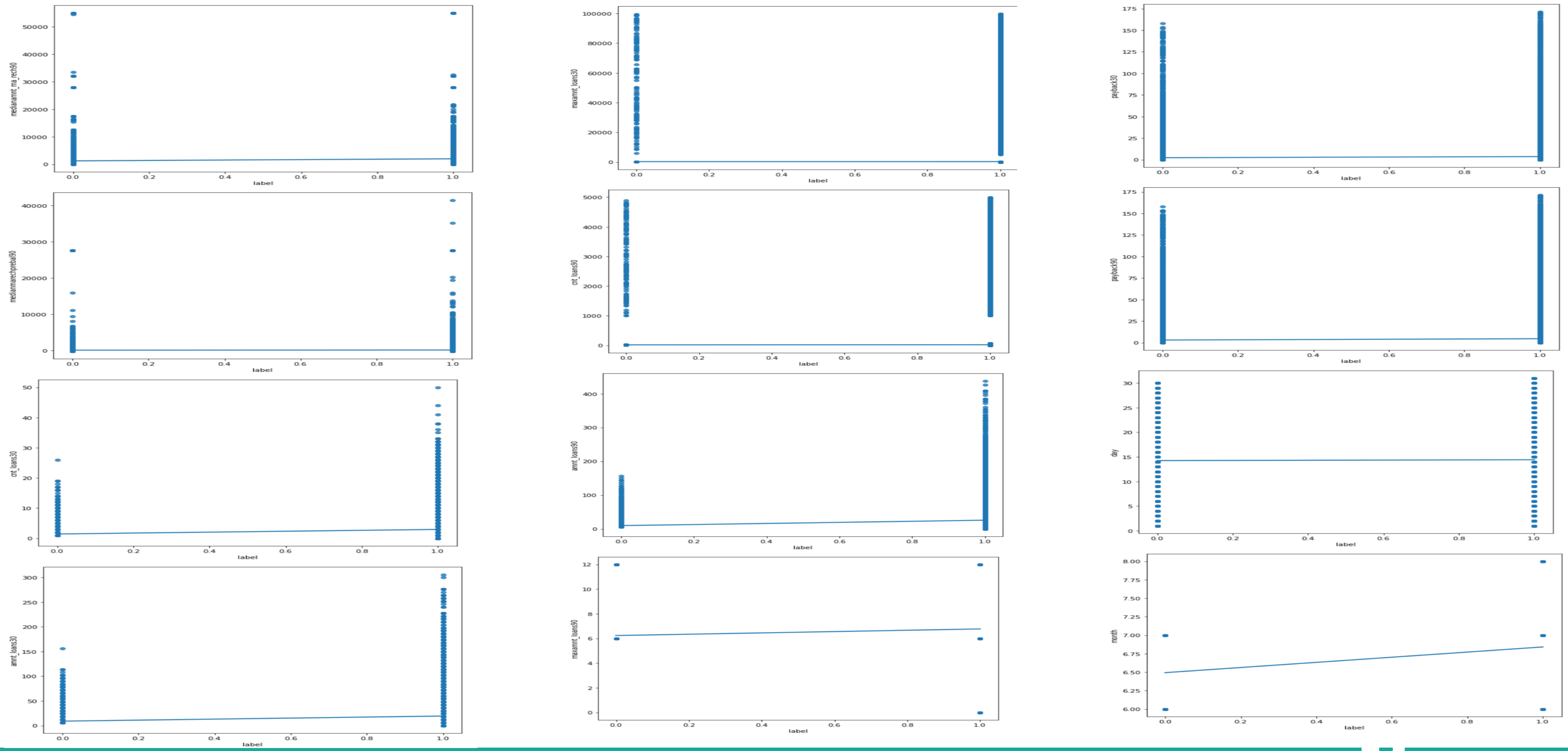




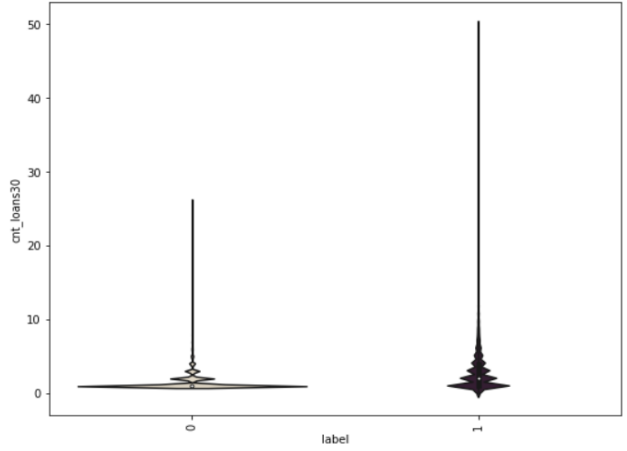


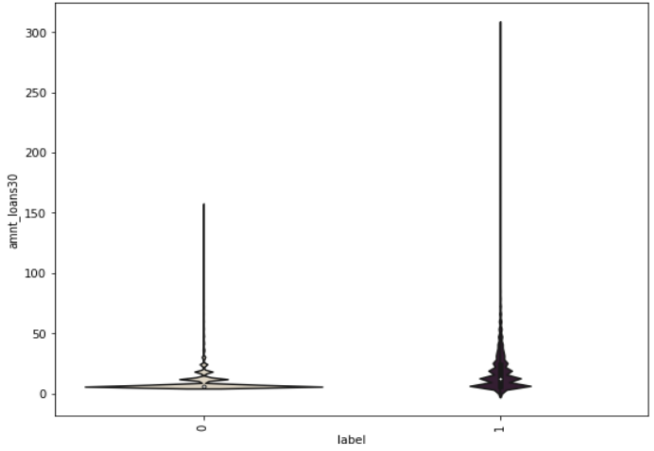


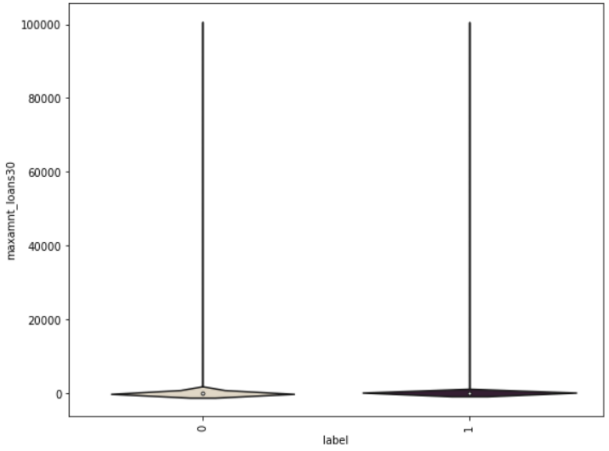
Bivariate Analysis: (Cont.)

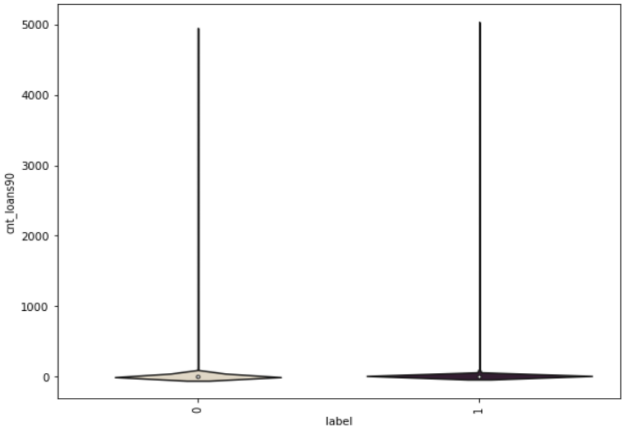


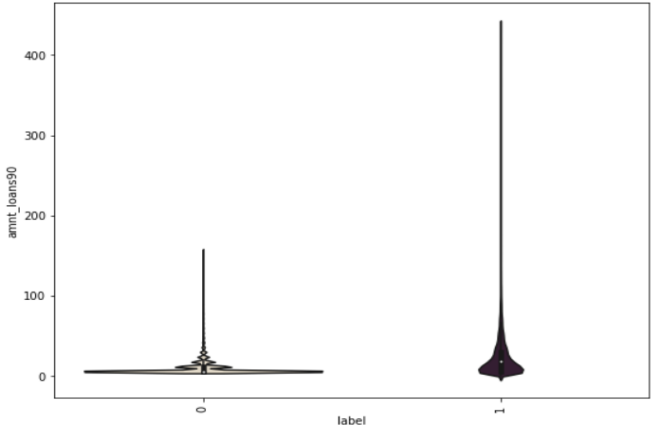
Bivariate Analysis: (Cont.)

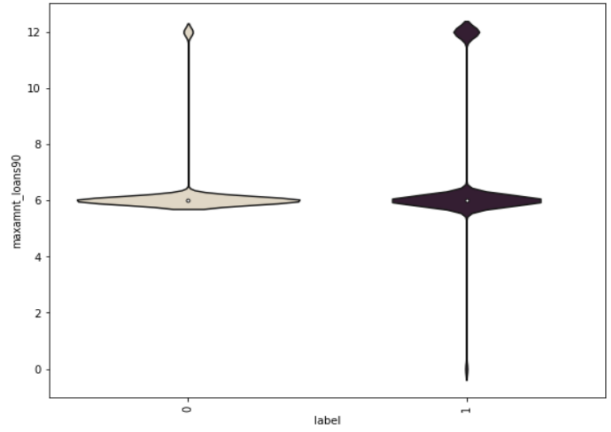




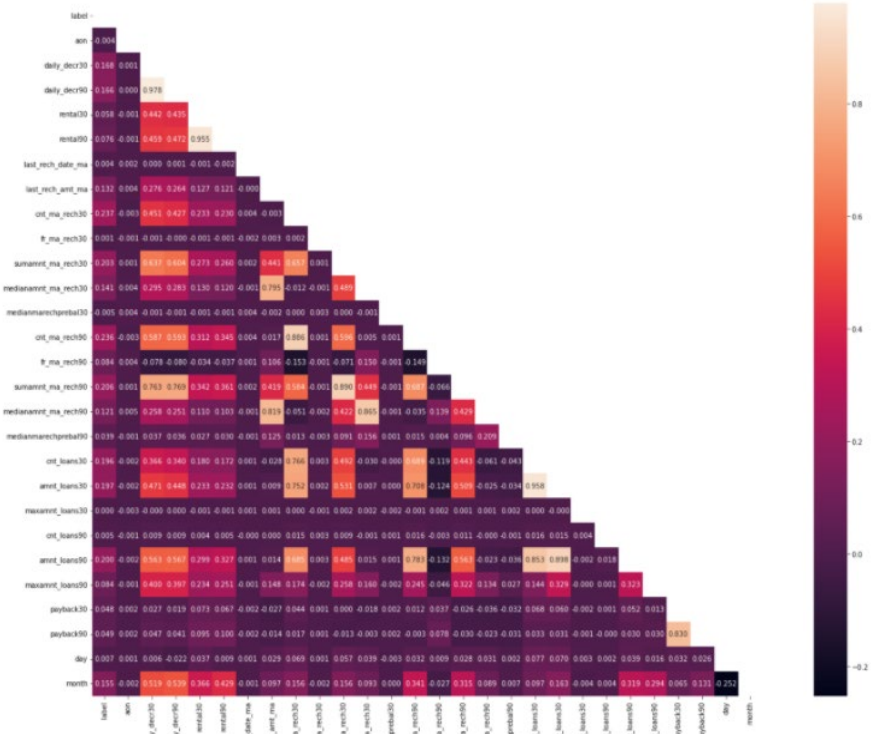


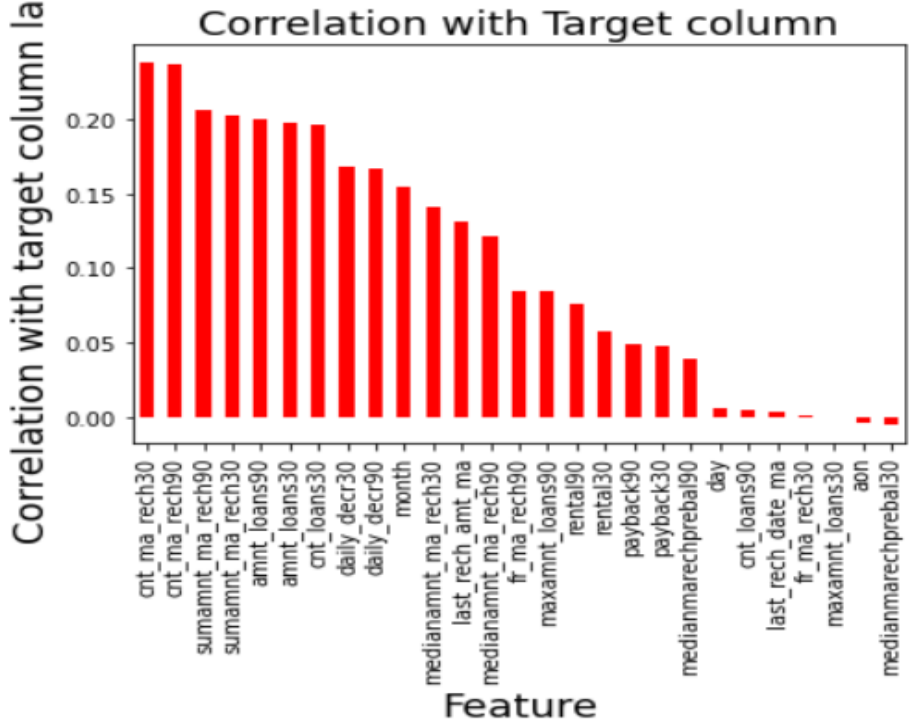






Multivariate Analysis:





Observations:

- We can see data imbalance in our target column which we will rectify later.
- We can Skewness towards the left, in columns aon, daily_dect30, daily_decr90, rental30, rental90, last_rech_date_ma, last_rech_amt_ma, cnt_ma_rech30, ft_ma_rech30, sunamnt_ma_rech30, medianamnt_ma_rech30, medianmarechprebal30, cnt_ma_rech90, fr_ma_rech90, sumamnt_ma_rech90, medianamnt_ma_rech90, medianmarechprebal90, cnt_loans30, amnt_loans30, maxamnt_loans30, cnt_loans90, amnt_loans90, payback30, payback90,.
- maxamnt_loans90 column is skewed towards the right which denotes median is more than mean.
- Here we can see that there are only 2 values present in our target column i.e. 1 and 0 and we can clearly see data imbalance in this column. we will balance the data using oversampling method.
- Here we can see Correlation of our target column with other columns present in the dataset.
- Here looking at plot of label and age on cellular network in days, we can say that defaults can happen even if the user is using services from longtime.
- Here correlation of target column label with columns daily_decr30, daily_decr90, rental30, rental90, last_rech_amt_ma, cnt_ma_rech30, sumamnt_ma_rech30, medianamnt_ma_rech30, cnt_ma_rech90, sumamnt_ma_rech90, medianamnt_ma_rech90, medianmarechprebel90, cnt_loans30, amnt_loans30, amnt_loans 90, we can say that There are less no. defaulters and most of the users paid there dues on time.

- Here we can see that the user who took loans in last 30 days paid back most of loans on time.
- Here we can observe that user who took loans in last 90 days was able to half of the loans on time while half users become defaulter as they couldn't pay back loan on time.
- Here we can observe that users paid most of the loans on time from total loans taken in 30 days, but they became defaulter in few loans.
- Here we can see that users paid most of the loans on time from the total loans taken in last 90 days, but users became defaulters in paying back few loans.
- Here we can observe that user was able to pay half of the amount of loan taken by him in last 30 days on time.
- Here we can observe that users paid all small amount loans on time, but were able to pay half of large amount loan on time.
- Here we can observe multicolinearity in some columns which we need to handle later.
- here we can observe that our target column label shares very little relation with all other columns present in the dataset.
- Here we can see that our target column label shares positive relation with all the columns except 3 columns.

Analysis:

- Here we have used dist plot for each univariate numerical features and it says that there is skewness in almost all columns.
- We used Pie plot for Target column label univariate analysis and we found that data is imbalance which we rectified later.
- And also for bivariate Analysis we have used Regression plot.
- In our analysis we found that in most features the count of non-defaulters is high compared to defaulters so , we can say that risk of micro credit default is less.

Steps used for Cleaning Data:

- ✓ In the dataset we did not found any null values there were many 0 values, but I found huge amount of outliers and very high skewness.
- ✓ To remove outliers we used percentile method. And for removing skewness we used yeo-johnson method.
- ✓ Then we used Standard Scaler method to Scale the data.
- ✓ As we found issue of Multicolinearity while Multivariate Analysis, we removed Multicolnearity using Variance Inflation Factor.
- ✓ While Univariate Analysis we found that our Target column was imbalanced, so we balanced the target column using SMOTE method in Oversampling. Then after Preparing our data we moved to model selection.

Model Selection:

✓Since Label was our target column and it was a Categorical column, so this particular problem was a Classification problem. And we used all Classification algorithms to build our model. We Tried multiple models and to avoid the confusion of overfitting we went through cross validation. Below are the list of Classification algorithms I have used in my project.By looking into the least difference of accuracy score and cross validation score I found **RandomForestClassifier** as a best model. Other models which i tried were:

- KNeighborsClassifier
- ExtraTreesClassifier
- GradientBoostingClassifier
- DecisionTreeClassifier

Model Accuracy Scores which we got:

1) KNeighborsClassifier:

```
knn=KNN()
knn.fit(x_train,y_train)
pred_k=knn.predict(x_test)
print("Accuracy Score: ", accuracy_score(y_test,pred_k))
print("Confusion Matrix: ", confusion_matrix(y_test,pred_k))
print(classification_report(y_test,pred_k))
```

```
Accuracy Score:  0.8951471483477044
Confusion Matrix:  [[54144   586]
 [10954 44375]]
              precision    recall  f1-score   support

           0       0.83      0.99      0.90     54730
           1       0.99      0.80      0.88     55329

    accuracy                           0.90    110059
   macro avg       0.91      0.90      0.89    110059
weighted avg       0.91      0.90      0.89    110059
```

- Here we can see that KNN is predicting score of 89.51%

2) ExtraTreesClassifier:

```
etc=ExtraTreesClassifier()
etc.fit(x_train,y_train)
pred_e=etc.predict(x_test)
print("Accuracy Score: ",accuracy_score(y_test,pred_e))
print("Confusion Matrix: ", confusion_matrix(y_test,pred_e))
print(classification_report(y_test,pred_e))
```

```
Accuracy Score:  0.9598578944020935
Confusion Matrix:  [[53417  1313]
 [ 3105 52224]]
              precision    recall  f1-score   support

           0       0.95      0.98      0.96     54730
           1       0.98      0.94      0.96     55329

    accuracy                           0.96    110059
   macro avg       0.96      0.96      0.96    110059
weighted avg       0.96      0.96      0.96    110059
```

- Here we have got an excellent accuracy score of 95.98% using Extra Tree Classifier.

5) Random Forest Classifier:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
pred_r=rfc.predict(x_test)
print("Accuracy Score: ",accuracy_score(y_test,pred_r))
print("Confusion Matrix: ",confusion_matrix(y_test,pred_r))
print(classification_report(y_test,pred_r))
```

```
Accuracy Score:  0.9523800870442217
Confusion Matrix:  [[52430  2300]
 [ 2941 52388]]
              precision    recall  f1-score   support

           0       0.95      0.96      0.95     54730
           1       0.96      0.95      0.95     55329

    accuracy                           0.95    110059
   macro avg       0.95      0.95      0.95    110059
weighted avg       0.95      0.95      0.95    110059
```

- Here we have got accuracy score of 95.23% using Random Forest Classifier

3) Gradient Boosting Classifier:

```
gbc=GradientBoostingClassifier()
gbc.fit(x_train,y_train)
pred_g=gbc.predict(x_test)
print('Accuracy Score: ',accuracy_score(y_test,pred_g))
print('Confusion Matrix: ',confusion_matrix(y_test,pred_g))
print(classification_report(y_test,pred_g))
```

```
Accuracy Score:  0.8982273144404365
Confusion Matrix:  [[50129  4601]
 [ 6600 48729]]
              precision    recall  f1-score   support

           0       0.88      0.92      0.90     54730
           1       0.91      0.88      0.90     55329

    accuracy                           0.90    110059
   macro avg       0.90      0.90      0.90    110059
weighted avg       0.90      0.90      0.90    110059
```

- Here we have got accuracy score of 89.82% using GradientBoostingClassifier.

4) Decision Tree Classifier:

```
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
pred_d=dtc.predict(x_test)
print("Accuracy Score: ",accuracy_score(y_test,pred_d))
print("Confusion Matrix: ",confusion_matrix(y_test,pred_d))
print(classification_report(y_test,pred_d))
```

```
Accuracy Score:  0.9099937306353865
Confusion Matrix:  [[50191  4539]
 [ 5367 49962]]
              precision    recall  f1-score   support

           0       0.90      0.92      0.91     54730
           1       0.92      0.90      0.91     55329

    accuracy                           0.91    110059
   macro avg       0.91      0.91      0.91    110059
weighted avg       0.91      0.91      0.91    110059
```

- Here we have got the accuracy score of 90.99% using DecisionTreeMatrix

Cross Validation Scores:

```
# cvs score of KNN
print(cross_val_score(knn,x,y,cv=5).mean())
```

```
0.900690177989065
```

```
# cvs score of ETC
print(cross_val_score(etc,x,y,cv=5).mean())
```

```
0.9650686313419883
```

```
# cvs score of GBC
print(cross_val_score(gbc,x,y,cv=5).mean())
```

```
0.8943145835725369
```

```
#cvs score of DTC
print(cross_val_score(dtc,x,y,cv=5).mean())
```

```
0.9103996968783878
```

```
#cvs score of RFC
print(cross_val_score(rfc,x,y,cv=5).mean())
```

```
0.9520883860292191
```

Hyper Parameter Tuning:

```
parameters={'criterion':['gini', 'entropy'],
            'n_estimators':[100,200,300],
            'max_features':['auto', 'sqrt', 'log2'],
            }
```

```
gcv=GridSearchCV(RandomForestClassifier(),parameters,cv=5)
```

```
gcv.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [100, 200, 300]})
```

```
gcv.best_params_
```

```
{'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 300}
```

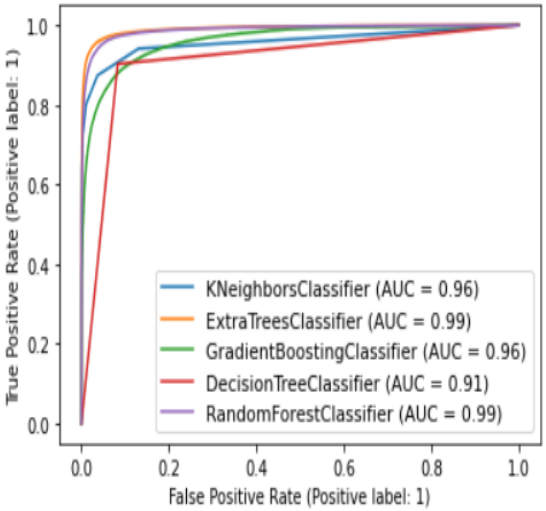
```
model=RandomForestClassifier(criterion='entropy',n_estimators=300 ,max_features=
model.fit(x_train,y_train)
pred=model.predict(x_test)
acc=accuracy_score(y_test,pred)
print('Accuracy Score: ',(accuracy_score(y_test,pred)*100))
print('Confusion Matrix: ',confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
Accuracy Score:  95.3461325289163
Confusion Matrix:  [[52470  2260]
 [ 2862 52467]]
              precision    recall  f1-score   support

           0       0.95      0.96      0.95     54730
           1       0.96      0.95      0.95     55329

    accuracy                           0.95    110059
   macro avg       0.95      0.95      0.95    110059
weighted avg       0.95      0.95      0.95    110059
```

- Here we have got excellent accuracy score of 95.34% using Random Forest Classifier. After HyperTuning we got improvement of 0.11%.

ROC-AUC Curve:

```
# lets import all required libraries
from sklearn import datasets
from sklearn import metrics
from sklearn import model_selection
from sklearn.metrics import plot_roc_curve
disp= plot_roc_curve(knn,x_test,y_test)
#ax_=Axes with confusion matrix
plot_roc_curve(etc,x_test,y_test, ax=disp.ax_)
plot_roc_curve(gbc,x_test,y_test, ax=disp.ax_)
plot_roc_curve(dtc,x_test,y_test, ax=disp.ax_)
plot_roc_curve(rfc,x_test,y_test, ax=disp.ax_)
plt.legend(prop={'size':11},loc='lower right')
plt.show()
```



- Here are the TOC Curves for all the models that i have used in model Selection and also all the AUC values can also be seen in the plot itself.

Saving The Model:

```
# saving the model as .pkl file
# importing library for sving the model
import joblib
joblib.dump(model,"MicroCredit_Defaulter.pkl")
```

```
['MicroCredit_Defaulter.pkl']
```

- Here we have successfully saved the model in .pkl format.

Loading Model and Predicting:

```
# loading the saved model
model=joblib.load("MicroCredit_Defaulter.pkl")

# predictions using our best model
prediction= model.predict(x_test)
prediction
```

```
array([1, 1, 0, ..., 0, 0, 0], dtype=int64)
```

Predictions:

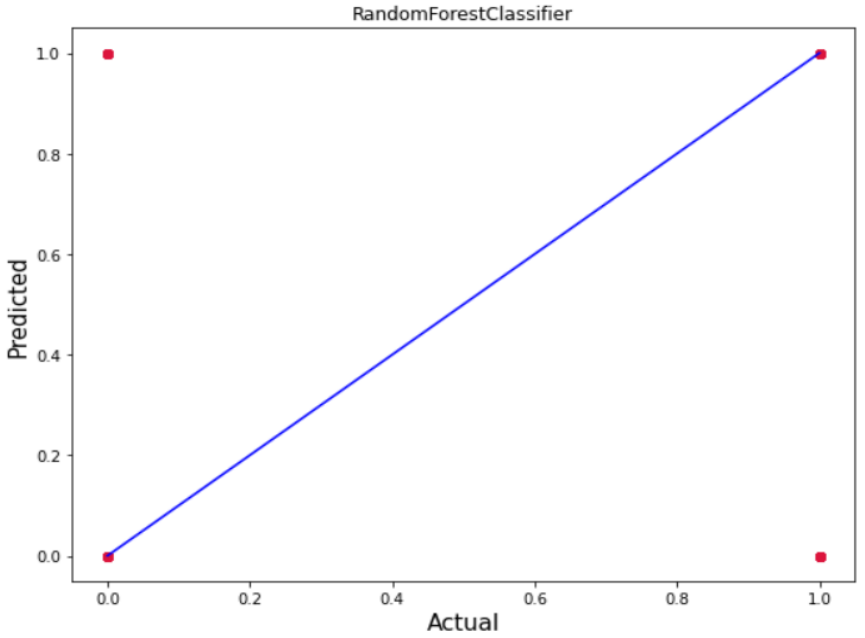
```
pd.DataFrame([model.predict(x_test)[:],y_test[:]],index=["Predicted","Actual"])
```

	0	1	2	3	4	5	6	7	8	9	...	110049	110050	110051	110052	110053	1100
Predicted	1	1	0	1	0	0	0	0	0	0	...	0	1	1	1	0	
Actual	1	1	0	1	0	0	0	0	0	0	...	0	1	1	1	1	

2 rows × 110059 columns

- Here we are getting most of the predicitions are accurate and actual values.

```
plt.figure(figsize=(9,7))
plt.scatter(y_test, prediction, c='crimson')
p1= max(max(prediction), max(y_test))
p2= min(min(prediction), min(y_test))
plt.plot([p1,p2],[p1,p2],'b-')
plt.xlabel('Actual', fontsize=15)
plt.ylabel('Predicted', fontsize=15)
plt.title("RandomForestClassifier")
plt.show()
```



- Here in the picture we can see Actual vs Predicted, Blue line refers to Actual Values and red dots are predicted values.

Conclusion:

- In this project report, we used machine learning algorithms to predict the Micro-Credit Defaulters. We used proper procedure to analyze the dataset and finding the correlation between the features.
- Here we selected the features which are correlated to each other and are independent in nature. Visualization helped us in understanding the data by graphical representation it made things easy for us to understand what data is trying to say.
- Data cleaning is one of the most important steps to remove unrealistic 0 values and columns which had morethan 90% 0 values.
- Using these feature we deployed 5 algorithms to find the best model and a hyper parameter tunning was done to the best model and we succeded in improvement of accuracy score.
- Then we saved the best model and predicted the label. Our model's performance felt good when we saw the predicted and actual values were almost same it felt really good observing good performance by our model.
- To conclude, the Project Micro Credit Defaulter , We hope this study will move a small step ahead in providing some methodological and empirical contributions to crediting institutes, and presenting an alternative approach to the valuation of defaulters.