Project Report:
“Micro- Credit Defaulter”

Submitted by: Amit Singh

ACKNOWLEDGMENT

It is a Great pleasure to express my Gratitude to Team FlipRobo, for giving me opportunity to work on a fantastic project, which helped me in improving my knowledge and coding skills.

Team FlipRobo also gave me opportunity to build PowerPoint Presentation and Project Report, which will help me to share steps taken while building the entire model. It will also help me with the story telling capabilities which are required to be a good Data Scientist. It is just a start I will try to convey my understanding of the project through this report.

INTRODUCTION

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.

Analytical Problem Framing

In this particular problem we have label as a target column and it has two classes 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. So clearly it is a binary classification problem and I have to use all classification algorithms for building the model. There was no null values in the dataset. Also, I observed some columns where I found more than 90% zero values so I decided to drop those columns. If I would have kept that columns then it would have affected my model’s performance. To get better insight on the features I have used plotting like distribution plot, pie plot and reg plot, violin plot. With these plotting I was able to understand the relation between the features in better manner. Also, I found huge amount of outliers and high skewness present in the dataset so I removed outliers using percentile method and I removed skewness using yeo-johnson method. I have used all the classification algorithms while building model then I tunned the best model and saved the best model. At last I have predicted the label using saved model.

Data Pre-processing:

- As a first step I have imported required libraries and I have imported the dataset using csv file provided by the company.
- Then we did all the statistical analysis like checking shape, unique, value counts, info and description etc…..
- Then while looking into the value counts, I found some columns with more than 90% zero values this creates skewness in the model and there are chances of getting model bias so I have dropped those columns with more than 90% zero values.
- While checking for null values I found no null values.
- I dropped Unnamed:0, msisdn and pcircle column as they were not needed.
- Next as a part of feature extraction I converted the pdate column to day, month and year.
- Also, I have dropped some columns when I tried to remove multicollinearity from the dataset using Variance Inflation Factor.
- Since we had all numerical columns so, I used dist plot to see the distribution of each column data.
- I have used violin plot and reg plot for each pair that shows the relation between label and independent features. Also we tried to observe whether the person pays back the loan within the date based on features.
- In maximum features relation with target, I observed non-defaulter count is high compared to defaulters.

Data Cleaning:

- In the dataset we did not find any null values there were many 0 values, but I found huge number 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 Multicollinearity while Multivariate Analysis, we removed Multicollinearity 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

Models Accuracy Scores:

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.

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

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

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}
```

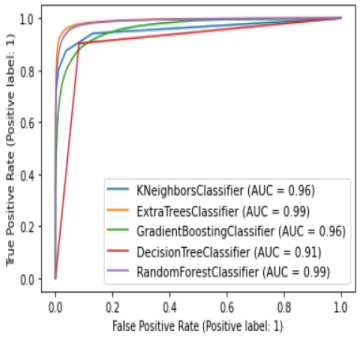
```
model=RandomForestClassifier(criterion='entropy',n_estimators=300 ,max_features=
model.fit(x_train,y_train)
pred=model.predict(x_test)
acc=accuracy_score(y_test,pred)
print('Accuracy Score: ',(accuracy_score(y_test,pred)*100))
print('Confusion Matrix: ',confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))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

```
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:

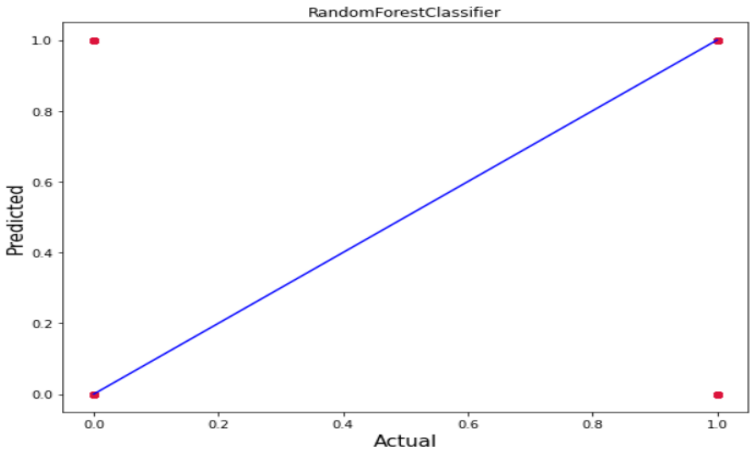
```
pd.DataFrame([model.predict(x_test)[:],y_test[:]],index=["Predicted","Actual"])
```

	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 more than 90% 0 values.
- ✧ Using these features, we deployed 5 algorithms to find the best model and a hyper parameter tunning was done to the best model and we succeeded 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.