



MICRO CREDIT DEFAULTER PREDICTION

Submitted by:

Neha Kamath

ACKNOWLEDGMENT

I would like to take this opportunity to thank my mentors at FlipRobo Technologies for their guidance and support in the completion of this project.

Index

Table of Contents

INTRODUCTION	4
Business Problem	4
Conceptual Background	4
Review of Literature	4
Motivation.....	5
ANALYTICAL PROBLEM REVIEW	6
Analytical Modeling	6
Data Sources	6
Data Pre-processing	7
Data Input-Output Relationships	8
Hardware and Software Tools Used	8
MODEL DEVELOPMENT AND EVALUATION	9
Problem Solving Approaches	9
Algorithms.....	9
Performance of Models	9
Key Metrics	16
Visualizations	17
Interpretations.....	20
CONCLUSION	13
Key Findings	21
Scope for Future.....	21

INTRODUCTION

- **Business Problem**

Our client, a telecom provider is collaborating with a micro-finance institution to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is classified as a defaulter if he does not payback the loaned amount within 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).

In order to improve the selection of customers for the credit, the telecom provider wants some predictions that could help them in further investment and improvement in selection of customers.

Our task is to build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan.

- **Conceptual Background**

Before we dive right into the project, it would be prudent to be acquainted with the domain related concepts such that not only do we have a clear understanding of the problem but also can focus our linear efforts on deriving a logical solution.

Given that client strategy is to run a budget operator model with better products at low prices to value conscious customers and knowing fully well the importance of communication to uplift low income families and poor customers, the client has tied up an MFI.

MFI (micro finance institution) 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. Today, microfinance is widely accepted as a poverty-reduction tool, representing \$70 billion in outstanding loans and a global outreach of 200 million clients.

- **Review of Literature.**

To this end, our client has provided us with sample customer data from their database. It contains information such as customer mobile number, age on the network, account balance, frequency of recharge, total amount of recharge, number of loans taken, amount of loan availed, payback time, payback date, telecom circle, status of loan- whether paid or defaulted, among other interesting details.

- **Motivation**

Our client is a fixed wireless telecommunications network provider. They have launched various products and have developed their 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. Thus, the need for collaborating with a micro finance institution to provide micro-credit on mobile balances to be paid back in 5 days.

Our primary focus and objective with this project is to help our client make an informed decision with respect to the selection of customers for providing this micro-credit such that there is a significant reduction in the percentage of defaulters.

Analytical Problem Review

- Analytical Modeling

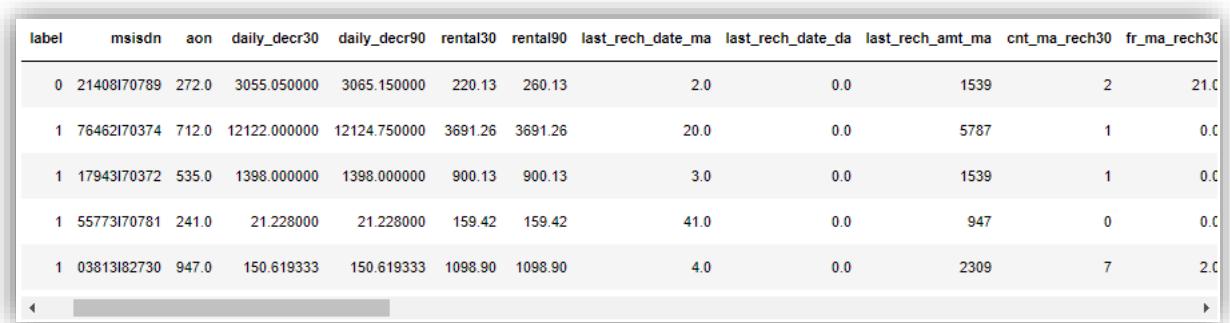
For the purpose of loan default prediction, we have a humongous amount of customer data that comprises of network usage, main and data account balance recharges, loan amounts, payback times and various other features that help us accurately model the information. In the course of modeling, we have applied various mathematical and statistical techniques that included handling skewness, treatment of outliers, data imbalance among others. Since the target variable is a Boolean (0:no-default and 1:default), we have applied and tested multiple classification algorithms to arrive at the optimal model alongwith hyper-parameter tuning to further improvise the model performance.

- Data Sources

Our primary source of data for this project has been the customer database provided by our client. It includes a total of 36 features and the number of records is 209593.

Label is our target, pcircle is a telecom circle and constant, date is provided, and many more features which are in numeric data formats. We will not require an encoder for this project.

Here is a glimpse of our dataset:



label	msisdn	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
0	21408170789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	2	21.0
1	76462170374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	1	0.0
1	17943170372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	1	0.0
1	55773170781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	0	0.0
1	03813182730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	7	2.0

And below you will find detailed descriptions of each of the variables:

Variable	Definition
label	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan[1:success, 0:failure]
msisdn	mobile number of user
aon	age on cellular network in days
daily_decr30	Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
daily_decr90	Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
rental30	Average main account balance over last 30 days
rental90	Average main account balance over last 90 days
last_rech_date_ma	Number of days till last recharge of main account
last_rech_date_da	Number of days till last recharge of data account
last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)
cnt_ma_rech30	Number of times main account got recharged in last 30 days
fr_ma_rech30	Frequency of main account recharged in last 30 days
sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
medianamnt_ma_rech30	Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90	Number of times main account got recharged in last 90 days
fr_ma_rech90	Frequency of main account recharged in last 90 days
sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days (in Indonesian Rupiah)
medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupiah)
medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah)
cnt_da_rech30	Number of times data account got recharged in last 30 days
fr_da_rech30	Frequency of data account recharged in last 30 days
cnt_da_rech90	Number of times data account got recharged in last 90 days
fr_da_rech90	Frequency of data account recharged in last 90 days
cnt_loans30	Number of loans taken by user in last 30 days
amnt_loans30	Total amount of loans taken by user in last 30 days
maxamnt_loans30	maximum amount of loan taken by the user in last 30 days
medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days
cnt_loans90	Number of loans taken by user in last 90 days
amnt_loans90	Total amount of loans taken by user in last 90 days
maxamnt_loans90	maximum amount of loan taken by the user in last 90 days
medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days
payback30	Average payback time in days over last 30 days
payback90	Average payback time in days over last 90 days
pcircle	telecom circle
pdate	date

• Data Preprocessing Done

The data pre-processing for this particular dataset required outlier checks and handling, skewness treatment as well as scaling. Since no categorical variables were present in the dataset for modeling, encoding was not required.

Outliers were present in most of the variables and so, the same were removed to a large extent using z-scores. Values with a z-score above 3 were eliminated.

The dataset was subject to skewness checks next and it was removed using the yeo-johnson method of the power transform function.

Since the values had widely different ranges, the dataset was scaled down to a standard scale using the Standard Scaler.

Data Imbalance, on account of unequal value counts in label, was treated using SMOTE oversampling technique.

- **Data Inputs- Logic- Output Relationships**

In this case, to derive the micro-credit loan applicant status (whether defaulter or non-defaulter), we are analyzing key factors such as customer tenure, daily balance used up, frequency of recharges, validity, total amount renewed, average recharge amount, amount and number of loans availed, payback times, all of which can be significant indicators or influencers of whether loans would be paid back or defaulted upon within the period of 5 days. For instance, we can use the previous payback times to make a fair judgement of whether the loan would be paid back in time or not. Frequency of recharges also has a good correlation with applicant status and can indicate whether the loan will be paid on time.

- **Hardware and Software Tools**

The libraries and packages we have used on this project are listed below:

- 📊 Data Processing- Numpy(numerical data wrangling), Pandas(data analysis)
- 📊 Data Visualization- Matplotlib, Seaborn (graphical representations)
- 📊 Data Preprocessing- Standard Scaler (scale variables in a standard range), Power Transform (treat skewness) from scikit-learn. SMOTE from imblearn library for treating data imbalance through oversampling.
- 📊 Modeling- Logistic Regression, Random Forest Classifier, Gradient Boosting Classifier, Decision Tree Classifier, K Neighbor Classifier and XGB Classifier (classification algorithms for model building).

Model Development and Evaluation

- Possible problem-solving approaches (methods)

The target 'label' contains a Boolean (0: Defaulter and 1: Non Defaulter). Hence, the logical approach to building a suitable prediction model is to use classification models such as logistic regression, RFCs, GBCs, DTCs, KNN etc.

- Testing of Identified Approaches (Algorithms)

- ✚ Logistic Regression,
- ✚ Random Forest Classifier,
- ✚ Gradient Boosting Classifier,
- ✚ Decision Tree Classifier,
- ✚ K Neighbor Classifier and
- ✚ XGB Classifier

- Performance of models

- ✚ **Logistic Regression:**

This is a machine learning algorithm for classification purposes. Using this, the probabilities describing the likely outcomes of a single trial are modelled using a logistic function.

A baseline model executed gave out an accuracy score of **78.06%**.

```
LR= LogisticRegression()
LR.fit(x_train,y_train)
predlr= LR.predict(x_test)
print(accuracy_score(y_test,predlr))
print(confusion_matrix(y_test,predlr))
print(classification_report(y_test,predlr))
```

0.7805790056519926				
[[27440 7286]				
[7971 26836]]				
	precision	recall	f1-score	support
0	0.77	0.79	0.78	34726
1	0.79	0.77	0.78	34807
accuracy			0.78	69533
macro avg	0.78	0.78	0.78	69533
weighted avg	0.78	0.78	0.78	69533

Post Hyper Parameter Tuning as below:

```
#Randomised Searchcv
param = {"penalty" : ['l1', 'l2', 'none'],
"C" : [0.00001,0.001,0.01,0.05,0.1,0.5,0.8,1,3,5,7,8,10,12,15,18,20,25,30,50,100,120,130,145,200,300,400,500,1000,5000]}

LR_rcv = RandomizedSearchCV(LR,param_distributions=param,verbose=5,n_jobs=-1,cv=5,n_iter=20)
LR_rcv.fit(x_train,y_train)
```

Best parameters obtained were penalty as l2 and C as 0.8, which resulted in model performance improving to **78.07%**

```
LR_rcv.best_params_

{'penalty': 'l2', 'C': 0.8}

LR_rcv_pred=LR_rcv.best_estimator_.predict(x_test)

print(accuracy_score(y_test,LR_rcv_pred))
print(confusion_matrix(y_test,LR_rcv_pred))
print(classification_report(y_test,LR_rcv_pred))
```

```
0.7806796772755382
[[27418  7308]
 [ 7942 26865]]
      precision    recall  f1-score   support

      0       0.78      0.79      0.78      34726
      1       0.79      0.77      0.78      34807

 accuracy          0.78
 macro avg          0.78
 weighted avg       0.78
```

🚀 Random Forest Classifier:

This classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and control over-fitting.

A single run of this gave an accuracy score of **94.95%**.

```
RF=RandomForestClassifier()
RF.fit(x_train,y_train)
predrf=RF.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
0.9494772266405879
[[33122 1604]
 [ 1909 32898]]
      precision    recall  f1-score   support

      0       0.95      0.95      0.95      34726
      1       0.95      0.95      0.95      34807

 accuracy          0.95
 macro avg          0.95
 weighted avg       0.95
```

Post Hyper Parameter Tuning as below:

```
#Randomised Searchcv
params = {"n_estimators":[100,400,800],"criterion":["gini", "entropy"]
          ,"max_depth":[3,7,8],"min_samples_split":[2,10,25],
          "max_features":["auto','sqrt','log2"],"min_samples_leaf":[1,5,6,7],"bootstrap":[True,False]}

RF_cv = RandomizedSearchCV(RF,param_distributions=params,n_iter=2,n_jobs=-1,cv=2,verbose=2)

RF_cv.fit(x_train,y_train)
```

Best parameters obtained are listed below, which resulted in model performance changing to **78.10%**

```
RF_cv.best_params_

{'n_estimators': 400,
 'min_samples_split': 2,
 'min_samples_leaf': 5,
 'max_features': 'auto',
 'max_depth': 3,
 'criterion': 'gini',
 'bootstrap': True}
```

```
RF_cv_pred=RF_cv.best_estimator_.predict(x_test)
```

```
print(accuracy_score(y_test,RF_cv_pred))
print(confusion_matrix(y_test,RF_cv_pred))
print(classification_report(y_test,RF_cv_pred))
```

```
0.7810967454302273
[[27266  7460]
 [ 7761 27046]]
```

	precision	recall	f1-score	support
0	0.78	0.79	0.78	34726
1	0.78	0.78	0.78	34807
accuracy			0.78	69533
macro avg	0.78	0.78	0.78	69533
weighted avg	0.78	0.78	0.78	69533



Gradient Boosting Classifier:

This is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

GBC initially generated an accuracy score of **89.77%**

```
GB = GradientBoostingClassifier()
GB.fit(x_train, y_train)
predgb = GB.predict(x_test)
print(accuracy_score(y_test,predgb))
print(confusion_matrix(y_test,predgb))
print(classification_report(y_test,predgb))
```

```
0.897746393798628
[[31538  3188]
 [ 3922 30885]]
```

	precision	recall	f1-score	support
0	0.89	0.91	0.90	34726
1	0.91	0.89	0.90	34807
accuracy			0.90	69533
macro avg	0.90	0.90	0.90	69533
weighted avg	0.90	0.90	0.90	69533

Post hyper parameter tuning:

```
#Randomized Searchcv

parameters = {"n_estimators":[100,500,900],"learning_rate":[0.01,0.3,0.7],
              ,"max_depth":[3,7,12],"min_samples_split":[2,12,20],
              "max_features":["auto','sqrt'],'min_samples_leaf':[1,5,8]}

GB_cv = RandomizedSearchCV(GB,param_distributions=parameters,n_iter=2,n_jobs=-1,cv=2,verbose=2)

GB_cv.fit(x_train,y_train)
```

Best parameters obtained are listed below, which resulted in model performance improving significantly to **94.65%**

```
GB_cv.best_params_
{'n_estimators': 900,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 3,
 'learning_rate': 0.7}
```

```
GB_cv_pred=GB_cv.best_estimator_.predict(x_test)
```

```
print(accuracy_score(y_test,GB_cv_pred))
print(confusion_matrix(y_test,GB_cv_pred))
print(classification_report(y_test,GB_cv_pred))
```

```
0.9465146045762444
[[32469 2257]
 [ 1462 33345]]
```

		precision	recall	f1-score	support
	0	0.96	0.94	0.95	34726
	1	0.94	0.96	0.95	34807
	accuracy			0.95	69533
	macro avg	0.95	0.95	0.95	69533
	weighted avg	0.95	0.95	0.95	69533



Decision Tree Classifier:

Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.

```
dt= DecisionTreeClassifier()
dt.fit(x_train,y_train)
prededt= dt.predict(x_test)
print(accuracy_score(y_test,prededt))
print(confusion_matrix(y_test,prededt))
print(classification_report(y_test,prededt))
```

```
0.9056850703982282
[[31735 2991]
 [ 3567 31240]]
```

		precision	recall	f1-score	support
	0	0.90	0.91	0.91	34726
	1	0.91	0.90	0.91	34807
	accuracy			0.91	69533
	macro avg	0.91	0.91	0.91	69533
	weighted avg	0.91	0.91	0.91	69533

A simple run gave out an accuracy score of **90.57%**

After hyper parameter tuning as per below code:

```
#Randomized Searchcv
params = {"max_depth" : [3,7,8,12] , "max_features" : list(range(1,7)), "min_samples_leaf" : [1,5,7,8],
          "criterion" : ["gini", "entropy"] }

DT_cv = RandomizedSearchCV(dt,param_distributions=params,n_iter=25,n_jobs=-1,cv=8,verbose=2)
DT_cv.fit(x_train,y_train)
```

Best parameters were obtained as listed below, which changed to accuracy to **84.54%**.

DT_cv.best_params_					
{ 'min_samples_leaf': 8, 'max_features': 4, 'max_depth': 12, 'criterion': 'entropy'}					
DT_cv_pred=DT_cv.best_estimator_.predict(x_test)					
print(accuracy_score(y_test,DT_cv_pred)) print(confusion_matrix(y_test,DT_cv_pred)) print(classification_report(y_test,DT_cv_pred))					
0.8453683862338746 [[28647 6079] [4673 30134]]					
	precision	recall	f1-score	support	
0	0.86	0.82	0.84	34726	
1	0.83	0.87	0.85	34807	
accuracy			0.85	69533	
macro avg	0.85	0.85	0.85	69533	
weighted avg	0.85	0.85	0.85	69533	



K Neighbor Classifier:

K-nearest neighbors (k-NN) is a pattern recognition algorithm that uses training datasets to find the *k* closest relatives in future examples. Initial run of this classifier resulted in accuracy of **89.49%**

knn = KNeighborsClassifier(n_neighbors = 5) knn.fit(x_train, y_train) predknn = knn.predict(x_test) print(accuracy_score(y_test,predknn)) print(confusion_matrix(y_test,predknn)) print(classification_report(y_test,predknn))					
0.8948988250183366 [[34270 456] [6852 27955]]					
	precision	recall	f1-score	support	
0	0.83	0.99	0.90	34726	
1	0.98	0.80	0.88	34807	
accuracy			0.89	69533	
macro avg	0.91	0.90	0.89	69533	
weighted avg	0.91	0.89	0.89	69533	

Post parameter tuning, the score went up to **93.72%**

```
#Randomized Searchcv

params = {"leaf_size":list(range(1,2)), "n_neighbors":list(range(1,2)), "p":[1,2]}

KN_cv = RandomizedSearchCV(knn,param_distributions=params,cv=2)

KN_cv.fit(x_train,y_train)
```

Best parameters as below:

```
KN_cv.best_params_

{'p': 1, 'n_neighbors': 1, 'leaf_size': 1}

KN_cv_pred=KN_cv.best_estimator_.predict(x_test)

print(accuracy_score(y_test,KN_cv_pred))
print(confusion_matrix(y_test,KN_cv_pred))
print(classification_report(y_test,KN_cv_pred))

0.9372240518890311
[[34414  312]
 [ 4053 30754]]
      precision    recall  f1-score   support

      0       0.89      0.99      0.94       34726
      1       0.99      0.88      0.93       34807

 accuracy          0.94
 macro avg         0.94
 weighted avg      0.94
```

XGB Classifier:

This is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. XGBoost provides a parallel tree boosting that solve many data science problems in a fast and accurate way.

This classifier gives us an accuracy score of **94.79%**.

```
XGB = XGBClassifier()
XGB.fit(x_train, y_train)
predxg = XGB.predict(x_test)
print(accuracy_score(y_test,predxg))
print(confusion_matrix(y_test,predxg))
print(classification_report(y_test,predxg))

[22:30:56] WARNING: C:/Users/Administrator/workspace/xg
0, the default evaluation metric used with the objectiv
t eval_metric if you'd like to restore the old behavior
0.9478952439848705
[[32689  2037]
 [ 1586 33221]]
      precision    recall  f1-score   support

      0       0.95      0.94      0.95       34726
      1       0.94      0.95      0.95       34807

 accuracy          0.95
 macro avg         0.95
 weighted avg      0.95
```

It was hyper parameter tuned:

```
#Randomised Searchcv

paramsdist = {"n_estimators": [100, 300, 700], "learning_rate": [0.001, 0.02, 0.7],
              "max_depth": [3, 5, 7, 10, 12, 15], "gamma": [0.001, 0.1, 3], "reg_alpha": [0, 0.1, 0.7],
              "reg_lambda": [1, 10, 80, 150]}

XG_cv = RandomizedSearchCV(XGB, param_distributions=paramsdist, n_jobs=-1, n_iter=2, cv=2, verbose=5)

XG_cv.fit(x_train, y_train)
```

To derive an even better score of **94.90%**

```
XG_cv.best_params_

{'reg_lambda': 10,
 'reg_alpha': 0,
 'n_estimators': 100,
 'max_depth': 15,
 'learning_rate': 0.7,
 'gamma': 0.001}

XG_cv_pred=XG_cv.best_estimator_.predict(x_test)

print(accuracy_score(y_test,XG_cv_pred))
print(confusion_matrix(y_test,XG_cv_pred))
print(classification_report(y_test,XG_cv_pred))

0.9490026318438727
[[32711 2015]
 [ 1531 33276]]
      precision    recall  f1-score   support

     0       0.96      0.94      0.95      34726
     1       0.94      0.96      0.95      34807

 accuracy          0.95
 macro avg         0.95      0.95      0.95      69533
weighted avg         0.95      0.95      0.95      69533
```

- Key Metrics

- ✚ **Confusion Matrix:**

It is the easiest way to measure performance for classification problems where target can be types of classes. Simply put, it is a two dimensional table for Actuals and Predictions.

Confusion Matrix		
	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

TP and TN is when Predictions meet Actuals.
FP and FN is when they don't.

- ✚ **Accuracy Score:**

Accuracy Score may be defined as the simple ratio between the number of correctly classified points (predictions) to the total number of points (predictions) and is the most commonly used performance metric for classification of algorithms.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

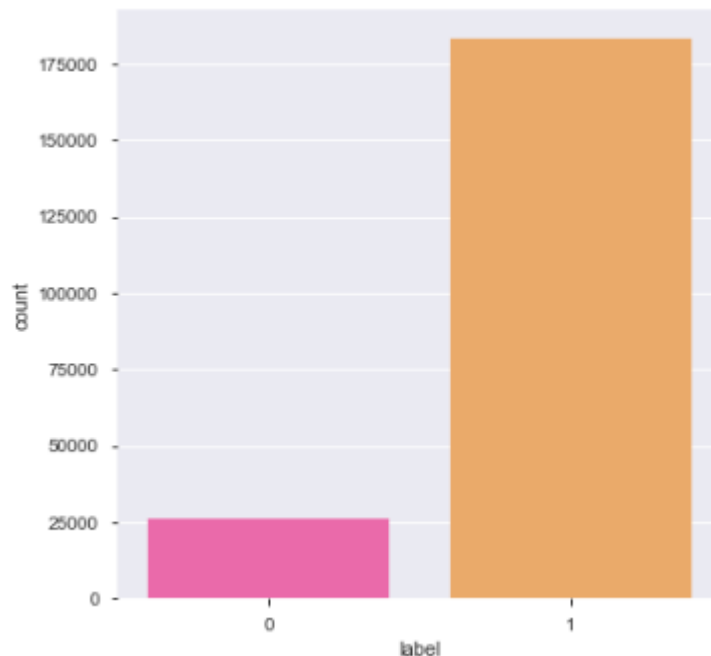
- ✚ **Classification Report:**

Using the classification report can give one a quick intuition of how the model is performing. Recall pits the number of examples your model labeled as Class X against the total number of examples of Class X. Precision is the percentage of examples your model labeled as Class A which actually belonged to Class A (true positives against false positives), and f1-score is an average of precision and recall.

$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN} \\ F1 &= \frac{2 \times precision \times recall}{precision + recall} \end{aligned}$$

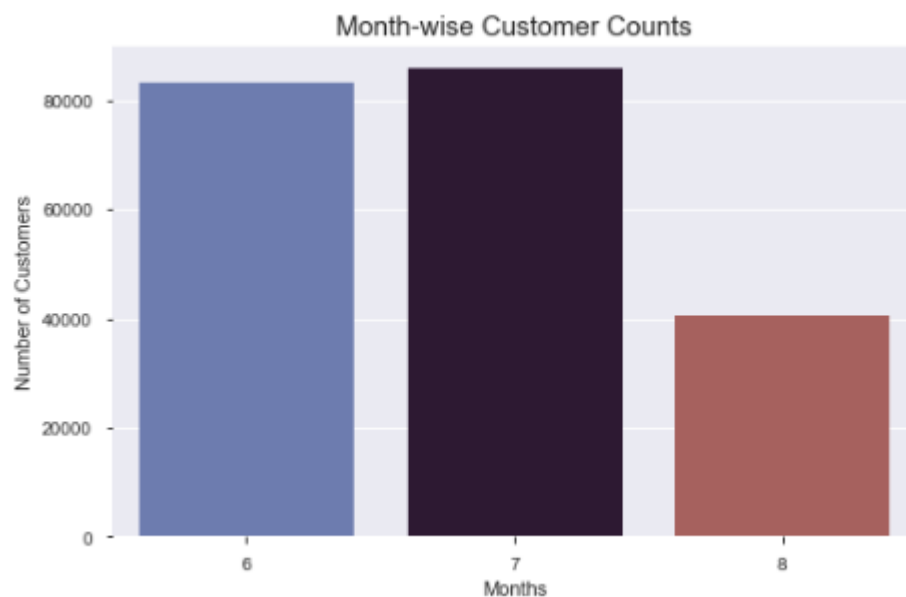
- Visualizations

- **Univariate Analysis with Target Distribution using countplot**



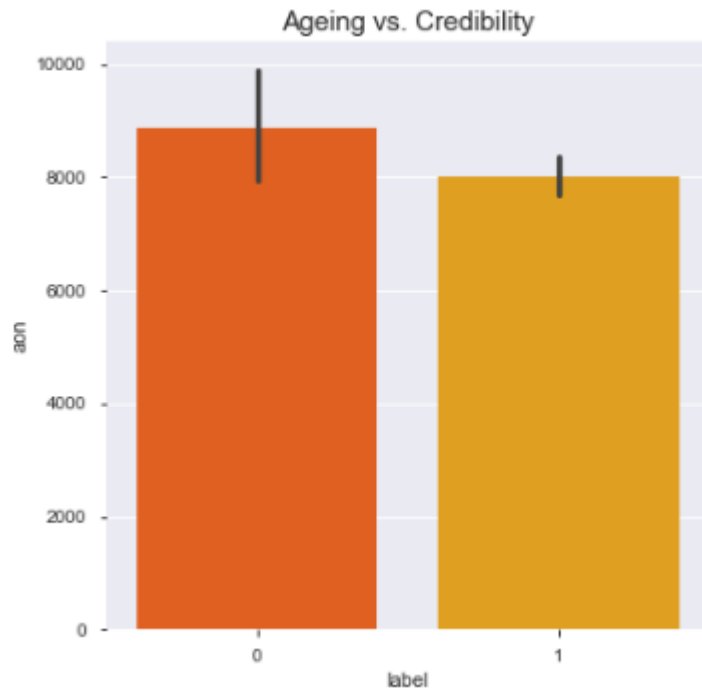
From this plot, it is easy to note on the imbalance in data between the Defaulter class and the Non-defaulter class. We have handled this class imbalance before the data was modelled.

- **Bi-variate analysis with Month-wise Customer Plot**



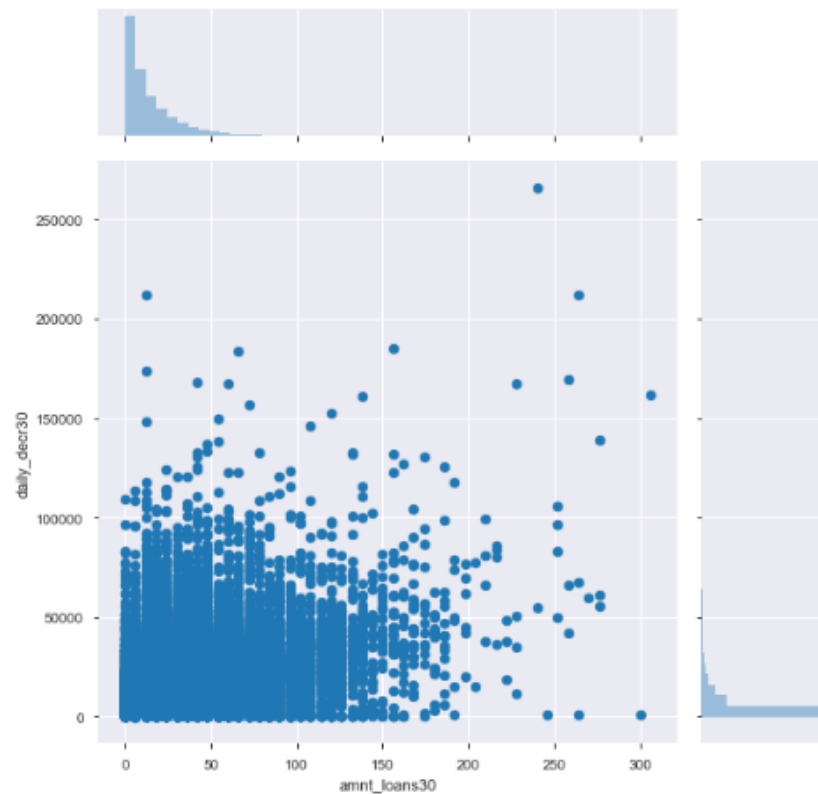
We see more customer footprint in the months of July and June as compared to August. This likely indicates a monthly trend where more users would need credit.

Bi-variate analysis with Age on Network and Loan status



This helps us plot a relation between customer age on network and their likelihood for defaults. The more the time a customer has spent on the same network, lesser is the chance of a default. Our client can give more preference to loyal customers for giving out credit.

Bivariate Analysis with Total loan amount and Daily balance used-Jointplot



Here, we can see a faint linear relationship between the two.

Multi-variate analysis with Density Plot



The density plot for all variables shows the distribution of data in each. We can observe a significant level of skewness in the data, which we have treated before building the models.

- Interpretation of the Results

We have tested various machine learning models to predict whether the mobile network customer would default on the micro-credit loan or pay it back within the stipulated period.

Let us take a look at the performance to find out the best:

Algorithms Tested	Key Metrics of the Hyper Parameter Tuned Model			
	Accuracy Score	Precision (0 1)	Recall (0 1)	F1-score (0 1)
Logistic Regression	0.7807	0.78 0.79	0.79 0.77	0.78 0.78
Random Forest Classifier	0.7811	0.78 0.78	0.79 0.78	0.78 0.78
Decision Tree Classifier	0.8554	0.86 0.83	0.82 0.87	0.84 0.85
K Neighbors Classifier	0.9372	0.89 0.99	0.99 0.88	0.94 0.93
Gradient Boosting Classifier	0.9465	0.96 0.94	0.94 0.96	0.95 0.95
XGB Classifier	0.9490	0.96 0.94	0.94 0.96	0.95 0.95

From the above table, we can observe that Logistic Regression and Random Forest Classifier models perform similarly where key metrics are concerned. Decision Tree classifier and K Neighbors Classifier models come next with an improved performance.

However, the models with the best accuracy score and f1-score are Gradient Boosting Classifier and XGB Classifier models at 95% f1-score.

Hence, we will be saving the **XGB classifier model** that is hyper-parameter tuned with optimal parameters as below for the purpose of predicting the micro-credit defaulters.

```
{'reg_lambda': 10,  
  'reg_alpha': 0,  
  'n_estimators': 100,  
  'max_depth': 15,  
  'learning_rate': 0.7,  
  'gamma': 0.001}
```

CONCLUSION

- Key Findings and Conclusions of the Study

The key takeaway from this study has been that what you feed the machine learning model is more important than the model or algorithm itself. Of its own accord, it cannot generate predictions with great accuracy. The GIGO (Garbage In Garbage Out) concept is especially true with machine learning. It is absolutely critical, therefore, that special attention is paid to the cleaning of the raw data- treating missing values, handling outlier values, removing skewness, dropping features that are not useful, scaling down the values for the entire dataset, handling data imbalances and visualising to derive meaningful relations between different variables.

- Limitations of this work and Scope for Future Work

Since the model has been tested on this specific database alone, it would be helpful to also look at additional customer detail databases from different geographies and timeframes to thoroughly test the effectiveness and accuracy of the model we have built for this exercise.

We could also go a step further with this project analysis and introduce **a new and exciting way** to encourage customers to payback their loans well within the timeframe.

A system could be designed around **rewards** whereby the mobile network customer would earn a redeemable reward point or a bonus recharge or a one-day extension to his existing data plan, in lieu of paying back the loaned amount in the first two or three days, perhaps. This would not only make a dull, duty-bound activity of loan payback fun for consumers but also ensure most of them payback the loans within the 5 day period and that the bad loans/defaults reduce.

Such an experiment could be tested and tried on a limited number of consumers to see its workability and efficacy in **reducing defaults** and improving **the quality of loans** for the telecom service provider. To this end, I would like to take this experimental project further and turn it into a practical application that will be available for widespread use in all geographies. One that would benefit the low-income customers by giving them access to network services through micro-credit on mobile balances as well as help the telecom service providers continue this credit service by ensuring minimal defaults.