Micro-Credit Defaulter Model

We are building 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. 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.

Steps to make a model-:

Data importing and visualization

a) Import the data and importing the basic libraries such as numpy, pandas, matplotlib and seaborn.
b) df is new variable In which whole data is consisting.
c) Now we explore the data with its shape that is 65535 rows and 37 columns.
d) The name of columns in which the target column is named as ‘label’.

e) df.info() will let us know with the type of integer which makes us understand that msisdn, pcircle are object whereas pdate is in date and time format.

f) df[‘column name’].unique will let us know with the unique values in a particular column.

g) Description will give us certain functions such as the mean, median, standard deviation, 25th percentile, 75th percentile, minimum and maximum. There is quite high difference in 75th percentile and maximum value which shows quite high no. of outliers.

h) Now we can really visualize the data-: by creating their countplot, histogram and stripplot of target variable.

i) We can make a scatter plot of two columns together also.

Checking the Correlation

a) We can check the correlation between the columns first by putting all data in a new variable named as dfc. It will let us know with the correlation value between all the columns with each other.

b) After this we can plot a heatmap getting correlation within columns. But there are objects also within these columns so first we change the object datatype to integer.

c) Using Label encoder we can transform object to integer data.

```
from sklearn.preprocessing import LabelEncoder

LE=LabelEncoder()

df['msisdn']=LE.fit_transform(df['msisdn'])
```

d) Hence we transform pcircle, pdate and msisdn into integers.

e) Now we can again take out the correlations b/w all, also with a heat map.

f) After checking the correlation between the target variable with all other columns we find highest correlation is with cnt_ma_rech30 and cnt_ma_rech90 that is 0.23

EDA

a) Create a distribution plot in the form of subplots giving the distribution and density of data in all columns.

ZSCORE=: from scipy.stats import zscore

First find out the zscore value of complete data and also the threshold value. With help of this threshold value we can find the zscore value of particular rows and columns.
After this find IQR values. First find the 75tth and 25th percentile of the columns.

IQR=Q3-Q1
IQR=0.75-0.25

z=np.abs(zscore(df))
z
We tried to remove the outliers by applying zscore method but it gave

the loss of 23% hence outliers not removed.

Skewness-: First we find the skewness value among all columns.

```
df.skew()
```

the highest skewness is in column name medianmarechprebal90 but its graph is giving negative skewness hence while removing skewness it becomes Nan. We will not touch it.

```
from scipy.stats import skew
```

plot the distribution graph which shows the skewness graph of all columns.
Now we can remove the skewness among the graph by square method.

```
Ex-:  df['label']=np.sqrt(df['label'])
      skew(df['label'])
```

We will get a value showing amount of skewness removed.

Model Creation

First find out the **Random state**.

The Random state is 164.

Next step is **train test split.**

X_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=164)

Model building

We will build the model by passing the four model test-:

a) from sklearn.linear_model import **LogisticRegression**

```
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))
```

Giving accuracy percentage as 88%.

b) from sklearn.tree import **DecisionTreeClassifier**

```
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
preddt= dt.predict(x_test)
print(accuracy_score(y_test,preddt)
)
print(confusion_matrix(y_test,predd
t))
print(classification_report(y_test,pr
eddt))
```

Giving accuracy percentage as 85%.

c) from sklearn.ensemble import **RandomForestClassifier**

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
predrf= rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predr
f))
print(classification_report(y_test,pr
edrf))
```

Giving accuracy percentage as 91%.

d) from sklearn.svm **import SVC**

```
svc=SVC()
svc.fit(x_train,y_train)
predsvc = svc.predict(x_test)
print(accuracy_score(y_test,predsvc))
print(confusion_matrix(y_test,predsvc))
print(classification_report(y_test,predsvc))
```

cross validation score

cross validation score of LogisticRegression model= 0.87

cross validation score of DecisionTree model=0.76

cross validation score of RandomTreeForest model= 0.76

cross validation score of SVC model= 0.76

Hyper parameter tuning

```
from sklearn.model_selection import GridSearchCV
```

We selected the model RandomTreeForest for the hypertuning as it has the best accuracy score of 91%.

```
mod=RandomForestClassifier(criterion='entropy',max_depth=7,random_state=164)
```

```
mod.fit(x_train,y_train)
pred=mod.predict(x_train)
```

```
print(accuracy_score(y_train,pred)*100)
```

The model is running at an excellence of **90%** accuracy.