Algo:ai
Empowering Enterprise AI

Project Report

(By: Rashid Ali)

YOUR REPOSITORIES (GITHUB.COM)

WWW.LINKEDIN.COM/IN/RASHID-ALI-460785232

ali996958@gmail.com

Topic: Delinquency Telecom Model

Objective :

- Create a delinquency model which can 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
  (Label ‘1’ & ’0’)
- Find Enclosed the Data Description File and The Sample Data for the Modeling Exercise

Introduction :

Definition:

Delinquency is a condition that arises when an activity or situation does not occur at its scheduled (or expected) date i.e., it occurs later than expected

Use Case:

Many donors, experts, and microfinance institutions (MFI) have become convinced that using mobile financial services (MFS) is more convenient and efficient, and less costly, than the traditional high-touch model for delivering microfinance services. MFS becomes especially useful when targeting the unbanked poor living in remote areas. The implementation of MFS, though, 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. One of our Client in Telecom collaborates with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be delinquent if he deviates from the path of paying back the loaned amount within 5 days

Machine Learning problem:

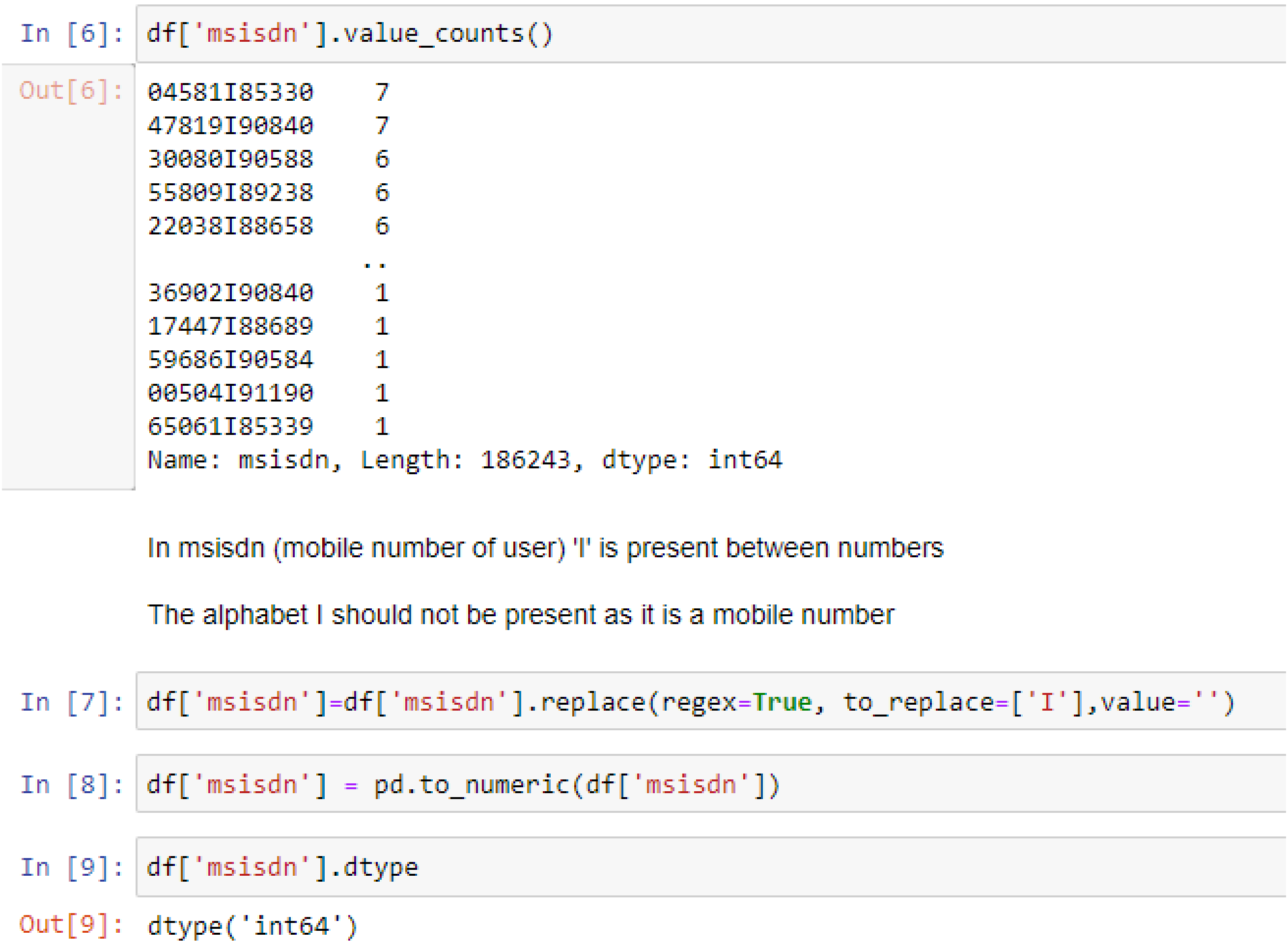
Create a delinquency model which can predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan (Label ‘1’ & ’0’) Basically a Binary Classification setup

Performance Metric:

- Log-loss (Since probabilities is our concern)
- Confusion matrix (Also want to check some precision and recalls)

Approach :

1. Firstly change the datatype of some Attribute like msisdn, pdate.



```
In [6]: df['msisdn'].value_counts()

Out[6]: 04581I85330    7
        47819I90840    7
        30080I90588    6
        55809I89238    6
        22038I88658    6
                      ..
        36902I90840    1
        17447I88689    1
        59686I90584    1
        00504I91190    1
        65061I85339    1
        Name: msisdn, Length: 186243, dtype: int64
```

In msisdn (mobile number of user) 'I' is present between numbers

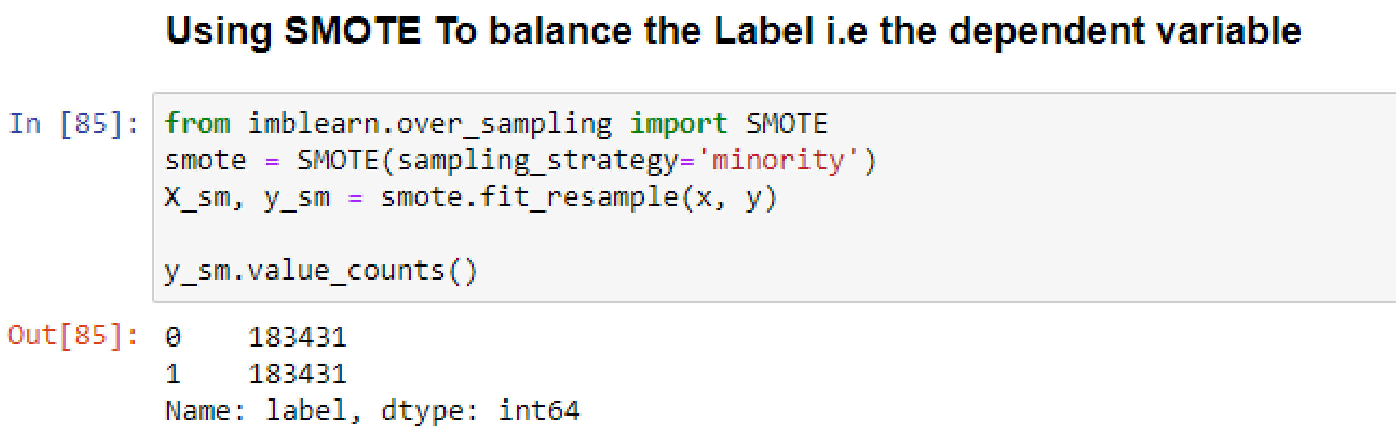
The alphabet I should not be present as it is a mobile number

```
In [7]: df['msisdn']=df['msisdn'].replace(regex=True, to_replace=['I'],value='')

In [8]: df['msisdn'] = pd.to_numeric(df['msisdn'])

In [9]: df['msisdn'].dtype

Out[9]: dtype('int64')
```

2. Checking Missing values,NaN, Duplicates etc.

Function: dataframe.info()

3. Checking Data Imbalances .



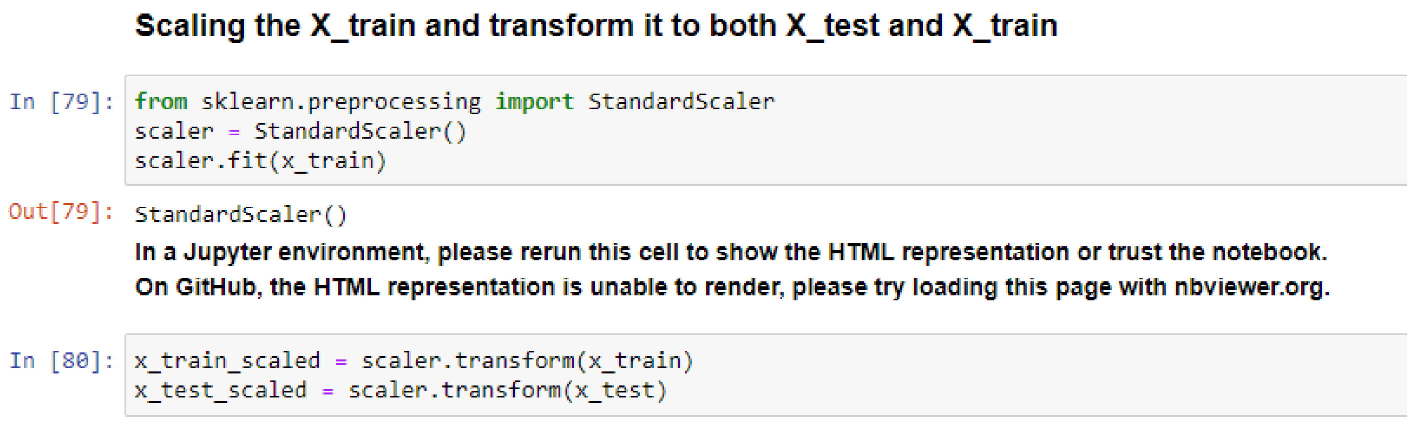
Using SMOTE To balance the Label i.e the dependent variable

```
In [85]: from imblearn.over_sampling import SMOTE
         smote = SMOTE(sampling_strategy='minority')
         X_sm, y_sm = smote.fit_resample(x, y)

         y_sm.value_counts()

Out[85]: 0    183431
         1    183431
         Name: label, dtype: int64
```

4. Checking Correlations among features

Function: Corr(dataframe)

5. Preprocessing the data.



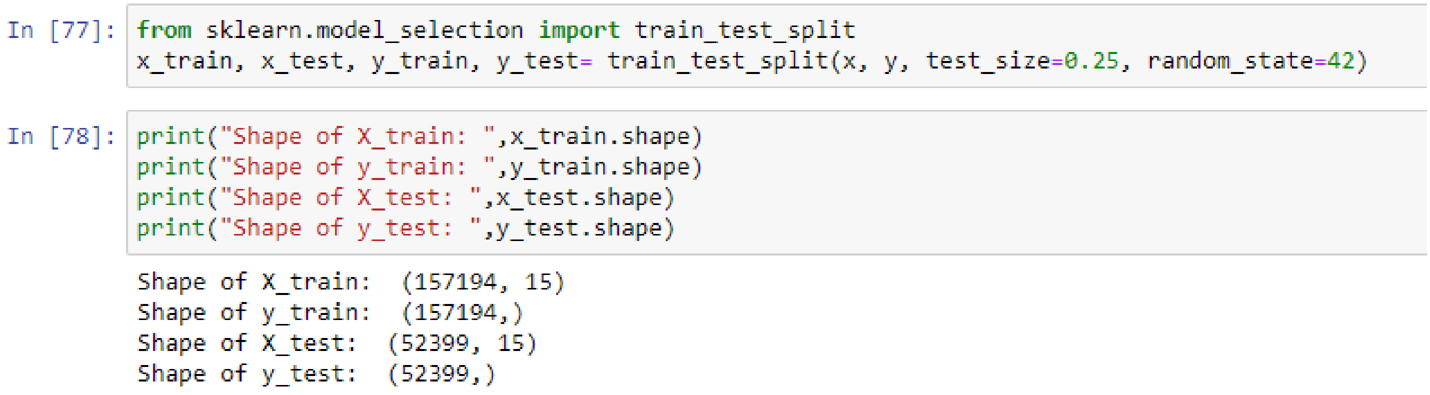
Scaling the X_train and transform it to both X_test and X_train

```
In [79]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         scaler.fit(x_train)

Out[79]: StandardScaler()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [80]: x_train_scaled = scaler.transform(x_train)
         x_test_scaled = scaler.transform(x_test)
```

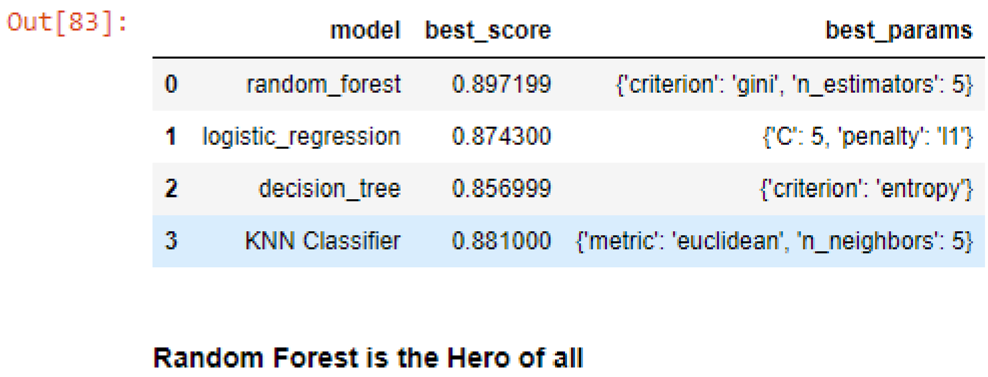
6. Train-Test Split.



```
In [77]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test= train_test_split(x, y, test_size=0.25, random_state=42)

In [78]: print("Shape of X_train: ",x_train.shape)
         print("Shape of y_train: ",y_train.shape)
         print("Shape of X_test: ",x_test.shape)
         print("Shape of y_test: ",y_test.shape)

         Shape of X_train:  (157194, 15)
         Shape of y_train:  (157194,)
         Shape of X_test:  (52399, 15)
         Shape of y_test:  (52399,)
```

7. Random Model Design for comparing it’s LogLoss with the ML models developed later on the dataset.

Before SMOTE :

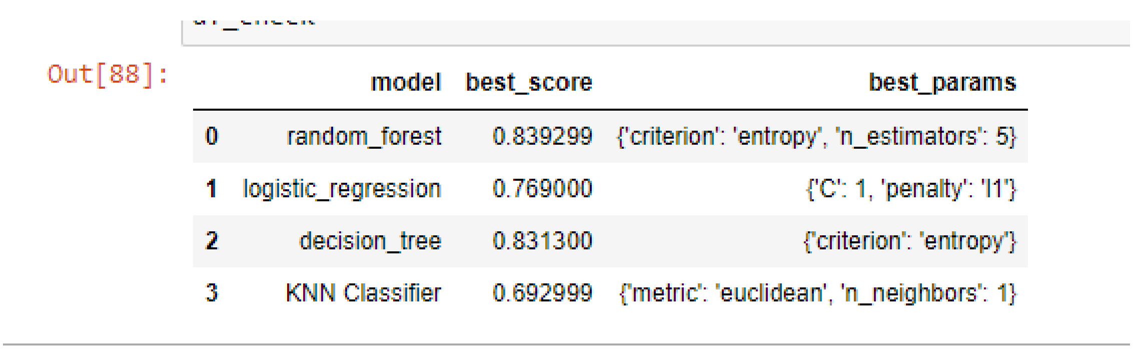


Out[83]:

	model	best_score	best_params
0	random_forest	0.897199	{'criterion': 'gini', 'n_estimators': 5}
1	logistic_regression	0.874300	{'C': 5, 'penalty': 'l1'}
2	decision_tree	0.856999	{'criterion': 'entropy'}
3	KNN Classifier	0.881000	{'metric': 'euclidean', 'n_neighbors': 5}

Random Forest is the Hero of all

AFTER SMOTE:

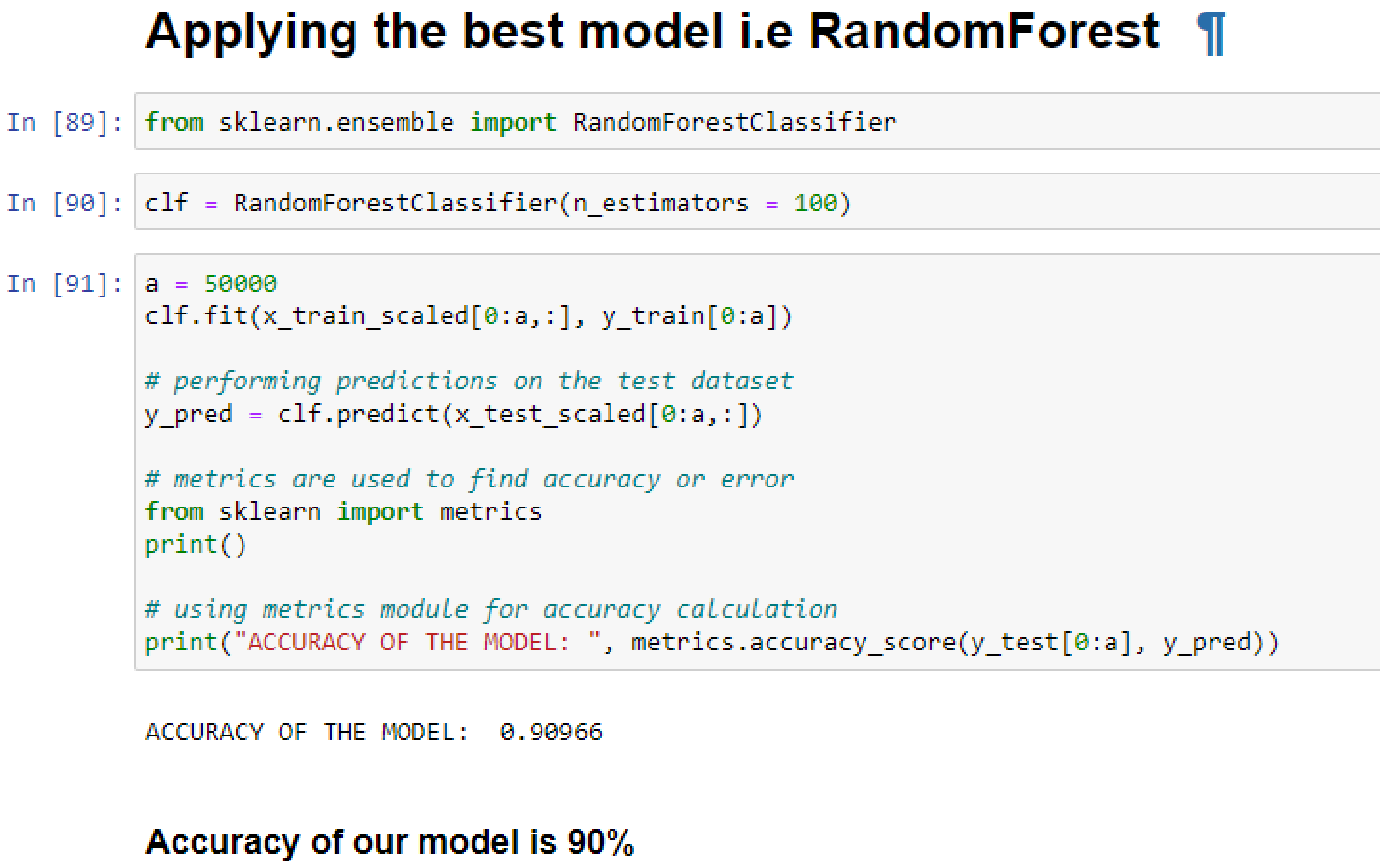


```
df_check
```

Out[88]:

	model	best_score	best_params
0	random_forest	0.839299	{'criterion': 'entropy', 'n_estimators': 5}
1	logistic_regression	0.769000	{'C': 1, 'penalty': 'l1'}
2	decision_tree	0.831300	{'criterion': 'entropy'}
3	KNN Classifier	0.692999	{'metric': 'euclidean', 'n_neighbors': 1}

8. MODELS USED and their results :



Applying the best model i.e RandomForest ¶

In [89]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [90]:
```
clf = RandomForestClassifier(n_estimators = 100)
```

In [91]:
```
a = 50000
clf.fit(x_train_scaled[0:a,:], y_train[0:a])

# performing predictions on the test dataset
y_pred = clf.predict(x_test_scaled[0:a,:])

# metrics are used to find accuracy or error
from sklearn import metrics
print()

# using metrics module for accuracy calculation
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test[0:a], y_pred))
```

```
ACCURACY OF THE MODEL:  0.90966
```

Accuracy of our model is 90%

END OF REPORT