In [15]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
data=pd.read_csv("C:\\Users\\eiti mittal\\Downloads\\data_F1score.csv")
```

In [3]:

```
data.head()
```

Out[3]:

	y_act	y_pred_random_forest	y_pred_logistic
0	1	0.639816	0.531904
1	0	0.490993	0.414496
2	1	0.623815	0.569883
3	1	0.506616	0.443674
4	0	0.418302	0.369532

The true values are binary and are classified in two classes:-0,1. F1 score is a classification metrics but here we are trying to compare binary true labels with continuous predictions

Random Forests and logistic regression are classification models that outputs probabilities. Therefore all these y_predicted values are actually the probabilities of y belonging to classes 0 or 1 and thus should lie between 0 <y_pred_prob <1

In [5]:

```
min(data['y_pred_random_forest']),max(data['y_pred_random_forest'])
```

Out[5]:

```
(0.17142946899999997, 0.930520528)
```

In [6]:

```
min(data['y_pred_logistic']),max(data['y_pred_logistic'])
```

Out[6]:

```
(0.00021747799999999998, 0.999762052)
```

The default threshhold used by scikit learn for binary probabilistic based models is 0.5, so we'll find actual predicted values using threshhold=0.5

In [9]:

```
data['y_pred_binary_random_forest']=(data['y_pred_random_forest'] >= 0.5).astype(int)
```

In [11]:

```
data['y_pred_binary_logistic']=(data['y_pred_logistic'] >= 0.5).astype(int)
```

In [12]:

```
data.head()
```

Out[12]:

	y_act	y_pred_random_forest	y_pred_logistic	y_pred_binary_random_forest	y_pred_binary_lo
0	1	0.639816	0.531904	1	
1	0	0.490993	0.414496	0	
2	1	0.623815	0.569883	1	
3	1	0.506616	0.443674	1	
4	0	0.418302	0.369532	0	

Calculate the TP, TN, FP, FN.

In [16]:

```
def comp_confmat(actual, predicted):

    classes = np.unique(actual) #extract the different classes
    matrix = np.zeros((len(classes), len(classes))) #initialize the confusion matrix with z

    for i in range(len(classes)):
        for j in range(len(classes)):

            matrix[i, j] = np.sum((actual == classes[i]) & (predicted == classes[j]))

    return matrix
```

In [28]:

```
conf_matrix_random_forest=comp_confmat(data['y_act'],data['y_pred_binary_random_forest'])
conf_matrix_random_forest
```

Out[28]:

```
array([[5519., 2360.],
       [2832., 5047.]])
```

In [79]:

```
conf_matrix_logistic=comp_confmat(data['y_act'],data['y_pred_binary_logistic'])
conf_matrix_logistic
```

Out[79]:

```
array([[5425., 2454.],
       [3600., 4279.]])
```

Compute Precision and recall for both the algorithms.

Take the mean of recall and precision values for both the classes as they have equal weight

In [40]:

```
recall_rf = (np.diag(conf_matrix_random_forest) / np.sum(conf_matrix_random_forest, axis =
precision_rf = (np.diag(conf_matrix_random_forest) / np.sum(conf_matrix_random_forest, axis
print("recall of random forest: ",recall_rf)
print("precision of random forest: ",precision_rf)
```

```
recall of random forest:  0.6705165630156111
precision of random forest:  0.6711307063452374
```

In [41]:

```
recall_lr = (np.diag(conf_matrix_logistic) / np.sum(conf_matrix_logistic, axis = 1)).mean()
precision_lr = (np.diag(conf_matrix_logistic) / np.sum(conf_matrix_logistic, axis = 0)).mea
print("recall of logistic regression: ",recall_lr)
print("precision of logistic regresion: ",precision_lr)
```

```
recall of logistic regression:  0.6158141896179719
precision of logistic regresion:  0.6183172722272119
```

A function to compute the F score for different values of beta:

In [65]:

```
def f_score(p,r,beta):
    score=(((beta**2)+1)*p*r)/(((beta**2)*p)+r)
    return score
```

Compare the F score at b##eta 0.5, 1, and 2

In [66]:

```
print(f_score(precision_rf,recall_rf,0.5))
print(f_score(precision_rf,recall_rf,1))
print(f_score(precision_rf,recall_rf,2))
```

```
0.671007787694254
0.6708234941173873
0.6706393017458937
```

In [67]:

```
print(f_score(precision_lr,recall_lr,0.5))
print(f_score(precision_lr,recall_lr,1))
print(f_score(precision_lr,recall_lr,2))
```

```
0.617815029154131
0.6170631925290971
0.6163131835425678
```

In [68]:

```
f1_score_rf=f_score(precision_rf,recall_rf,1)
f1_score_lr=f_score(precision_lr,recall_lr,1)
```

Model comparison

In [48]:

```
data['y_act'].value_counts()
```

Out[48]:

```
1    7879
0    7879
Name: y_act, dtype: int64
```

Therefore, this is an application of binary classifier on a balanced dataset

For a balanced class data set with equal importance/weight given to both classes, a model with a F1 score closer to 1 is a better model:

In [78]:

```
print("F1 score of random forest: ",f1_score_rf)
print("F1 score of logistic regression: ",f1_score_lr)
```

```
F1 score of random forest:  0.6708234941173873
F1 score of logistic regression:  0.6170631925290971
```

Best model- Random Forest

In [64]:

```
from sklearn.metrics import classification_report
```

In [61]:

```
print(classification_report(data['y_act'], data['y_pred_binary_random_forest']))
```

```
              precision    recall  f1-score   support

           0       0.66      0.70      0.68      7879
           1       0.68      0.64      0.66      7879

    accuracy                           0.67     15758
   macro avg       0.67      0.67      0.67     15758
weighted avg       0.67      0.67      0.67     15758
```

In [62]:

```
print(classification_report(data['y_act'], data['y_pred_binary_logistic']))
```

```
              precision    recall  f1-score   support

           0       0.60      0.69      0.64      7879
           1       0.64      0.54      0.59      7879

    accuracy                           0.62     15758
   macro avg       0.62      0.62      0.61     15758
weighted avg       0.62      0.62      0.61     15758
```