In [1]:

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
```

executed in 14.6s, finished 18:48:46 2022-01-05

In [2]:

```
bank_data=pd.read_csv('bank-full.csv',sep=';')
bank_data.head()
```

executed in 309ms, finished 18:49:11 2022-01-05

Out[2]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may

Initial investigation

In [3]:

```
bank_data.shape
```

Out[3]:

```
(45211, 17)
```

In [4]:

```
bank_data.dtypes
```

Out[4]:

```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
y            object
dtype: object
```

In [5]:

```
bank_data.isnull().sum()
```

Out[5]:

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

In [6]:

```
pd.set_option('max_column',None)
```

In [7]:

```
bank_data.head(20)
```

Out[7]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198
5	35	management	married	tertiary	no	231	yes	no	unknown	5	may	139
6	28	management	single	tertiary	no	447	yes	yes	unknown	5	may	217
7	42	entrepreneur	divorced	tertiary	yes	2	yes	no	unknown	5	may	380
8	58	retired	married	primary	no	121	yes	no	unknown	5	may	50
9	43	technician	single	secondary	no	593	yes	no	unknown	5	may	55

Number of features and records in the given data set is 17 and 45211 respectively

There is no null values in the data set

The categorical data can be converted into numeric data type by using encoder so that the model can learn the things more easily

Data preparation

In [8]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [9]:

```
bank_data['job'].value_counts()
```

Out[9]:

```
blue-collar      9732
management       9458
technician       7597
admin.           5171
services         4154
retired          2264
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: job, dtype: int64
```

In [10]:

```
bank_data['marital'].value_counts()
```

Out[10]:

```
married     27214
single      12790
divorced     5207
Name: marital, dtype: int64
```

In [11]:

```
bank_data['education'].value_counts()
```

Out[11]:

```
secondary    23202
tertiary     13301
primary       6851
unknown       1857
Name: education, dtype: int64
```

In [12]:

```
bank_data['default'].nunique(),bank_data['housing'].nunique(),bank_data['loan'].nunique()
```

Out[12]:

```
(2, 2, 2)
```

In [13]:

```
bank_data['contact'].unique()
```

Out[13]:

```
array(['unknown', 'cellular', 'telephone'], dtype=object)
```

In [14]:

```
bank_data['poutcome'].unique()
```

Out[14]:

```
array(['unknown', 'failure', 'other', 'success'], dtype=object)
```

In [15]:

```
bank_data[['job']]=le.fit_transform(bank_data[['job']])
bank_data[['marital']]=le.fit_transform(bank_data[['marital']])
bank_data[['education']]=le.fit_transform(bank_data[['education']])
bank_data[['default']]=le.fit_transform(bank_data[['default']])
bank_data[['contact']]=le.fit_transform(bank_data[['contact']])
bank_data[['poutcome']]=le.fit_transform(bank_data[['poutcome']])
bank_data[['y']]=le.fit_transform(bank_data[['y']])
bank_data
```

Out[15]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	dur
0	58	4	1	2	0	2143	yes	no	2	5	may	
1	44	9	2	1	0	29	yes	no	2	5	may	
2	33	2	1	1	0	2	yes	yes	2	5	may	
3	47	1	1	3	0	1506	yes	no	2	5	may	
4	33	11	2	3	0	1	no	no	2	5	may	
...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	9	1	2	0	825	no	no	0	17	nov	
45207	71	5	0	0	0	1729	no	no	0	17	nov	
45208	72	5	1	1	0	5715	no	no	0	17	nov	
45209	57	1	1	1	0	668	no	no	1	17	nov	
45210	37	2	1	1	0	2971	no	no	0	17	nov	

45211 rows × 17 columns

In [16]:

```
bank_data['housing']=le.fit_transform(bank_data['housing'])
bank_data['loan']=le.fit_transform(bank_data['loan'])
bank_data['month']=le.fit_transform(bank_data['month'])
bank_data
```

Out[16]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	dur
0	58	4	1	2	0	2143	1	0	2	5	8	
1	44	9	2	1	0	29	1	0	2	5	8	
2	33	2	1	1	0	2	1	1	2	5	8	
3	47	1	1	3	0	1506	1	0	2	5	8	
4	33	11	2	3	0	1	0	0	2	5	8	
...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	9	1	2	0	825	0	0	0	17	9	
45207	71	5	0	0	0	1729	0	0	0	17	9	
45208	72	5	1	1	0	5715	0	0	0	17	9	
45209	57	1	1	1	0	668	0	0	1	17	9	
45210	37	2	1	1	0	2971	0	0	0	17	9	

45211 rows × 17 columns

In [17]:

```
bank_data.dtypes
```

Out[17]:

```
age          int64
job          int32
marital      int32
education    int32
default      int32
balance      int64
housing      int32
loan         int32
contact      int32
day          int64
month        int32
duration     int64
campaign     int64
pdays        int64
previous     int64
poutcome     int32
y            int32
dtype: object
```

Model building

In [18]:

```
x=bank_data.iloc[:,:16]
y=bank_data['y']
```

In [19]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [20]:

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[20]:

```
((36168, 16), (9043, 16), (36168,), (9043,))
```

Model training

In [21]:

```
from sklearn.linear_model import LogisticRegression
log_model=LogisticRegression()
log_model.fit(x_train,y_train)
```

Out[21]:

```
LogisticRegression()
```

Model testing

In [22]:

```
y_pred_train=log_model.predict(x_train)
y_pred_test=log_model.predict(x_test)
```

Model Evaluation

In [23]:

```
from sklearn.metrics import classification_report,confusion_matrix,roc_auc_score,auc
```

In [24]:

```
print(classification_report(y_pred_train,y_train))
```

```
              precision    recall  f1-score   support

           0       0.98      0.90      0.94     34782
           1       0.18      0.55      0.27      1386

    accuracy                           0.89     36168
   macro avg       0.58      0.72      0.60     36168
weighted avg       0.95      0.89      0.91     36168
```

In [25]:

```
print(classification_report(y_pred_test,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.90      0.94      8696
           1       0.19      0.59      0.29       347

    accuracy                           0.89      9043
   macro avg       0.59      0.74      0.61      9043
weighted avg       0.95      0.89      0.92      9043
```

In [34]:

```
confusion_matrix_train=confusion_matrix(y_train,y_pred_train)
confusion_matrix_train
```

Out[34]:

```
array([[31310,   630],
       [ 3472,   756]], dtype=int64)
```

In [35]:

```
print(confusion_matrix(y_test,y_pred_test))
```

```
[[7839  143]
 [ 857  204]]
```

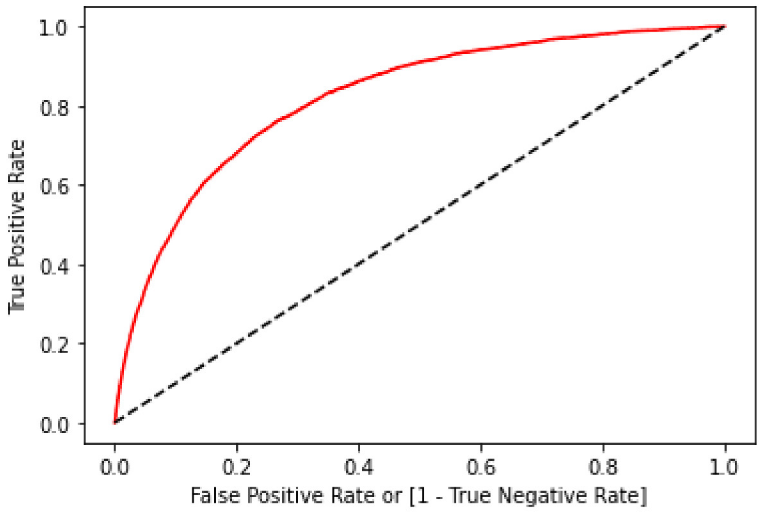
In [36]:

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

fpr, tpr, thresholds = roc_curve(y, log_model.predict_proba (x)[:,1])

auc = roc_auc_score(y_test, y_pred_test)
print(auc)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red', label='logit model ( area  = %0.2f)'%auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
0.5871780662947806
```

Out[36]:

```
Text(0, 0.5, 'True Positive Rate')
```



Imbalaced data

In [37]:

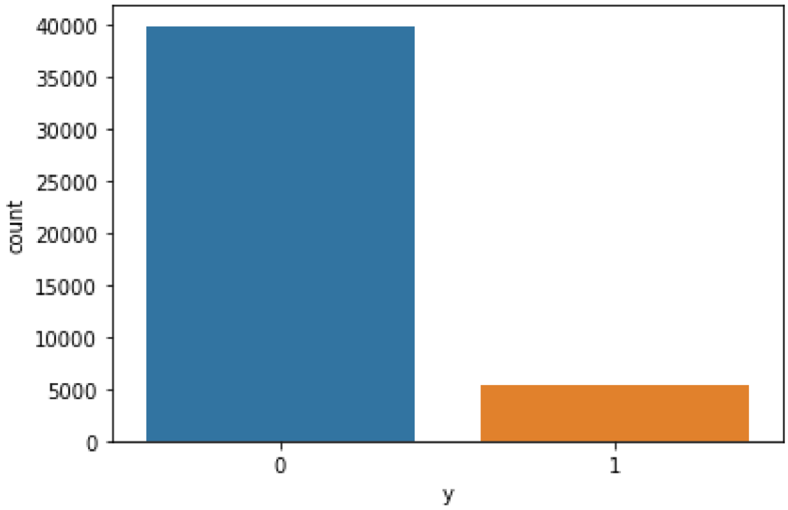
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [38]:

```
sns.countplot(bank_data['y'])
```

Out[38]:

```
<AxesSubplot:xlabel='y', ylabel='count'>
```



In [23]:

```
bank_data['y'].value_counts()
```

Out[23]:

```
0    39922
1     5289
Name: y, dtype: int64
```

The No of data available for the 'yes' category, is comparitively low than than 'No' category, this may affect the accuracy of the model because there is no balance of data between those categories

Hence there is a need of balancing the data.

Under sampling

Data preparation

In [44]:

```
count_0=bank_data[bank_data['y']==0]
count_1=bank_data[bank_data['y']==1]
```

In [45]:

```
count_0.shape,count_1.shape
```

Out[45]:

```
((39922, 17), (5289, 17))
```

In [46]:

```
under_count_0=count_0.sample(5289)
```

In [47]:

```
under_count_0.shape,count_1.shape
```

Out[47]:

```
((5289, 17), (5289, 17))
```

In [48]:

```
under_sample=pd.concat([under_count_0,count_1],axis=0)
under_sample
```

Out[48]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	dur
10753	58	4	1	0	0	1136	0	0	2	17	6	
8036	45	7	1	1	0	759	1	0	2	2	6	
2920	34	1	1	0	0	357	1	0	2	14	8	
28537	30	4	1	2	0	350	0	0	0	29	4	
18729	57	9	0	2	0	0	1	0	0	31	5	
...	...	...	...	...	...	...	...	...	...	...	...	
45204	73	5	1	1	0	2850	0	0	0	17	9	
45205	25	9	2	1	0	505	0	1	0	17	9	
45206	51	9	1	2	0	825	0	0	0	17	9	
45207	71	5	0	0	0	1729	0	0	0	17	9	
45208	72	5	1	1	0	5715	0	0	0	17	9	

10578 rows × 17 columns

Model building|training|testing|evaluation for undersampled data

In [49]:

```
under_x=under_sample.iloc[:,:16]
under_y=under_sample['y']
```

In [50]:

```
x_untrain,x_untest,y_untrain,y_untest=train_test_split(under_x,under_y,test_size=0.2)
```

In [51]:

```
under_model=LogisticRegression().fit(x_untrain,y_untrain)
```

In [52]:

```
y_pred_untrain=under_model.predict(x_untrain)
y_pred_untest=under_model.predict(x_untest)
```

In [53]:

```
print(confusion_matrix(y_untrain,y_pred_untrain))
```

```
[[3360  889]
 [1127 3086]]
```

In [55]:

```
print(confusion_matrix(y_untest,y_pred_untest))
```

```
[[817 223]
 [268 808]]
```

In [56]:

```
print(classification_report(y_untest,y_pred_untest))
```

```
              precision    recall  f1-score   support

           0       0.75      0.79      0.77      1040
           1       0.78      0.75      0.77      1076

    accuracy                           0.77      2116
   macro avg       0.77      0.77      0.77      2116
weighted avg       0.77      0.77      0.77      2116
```

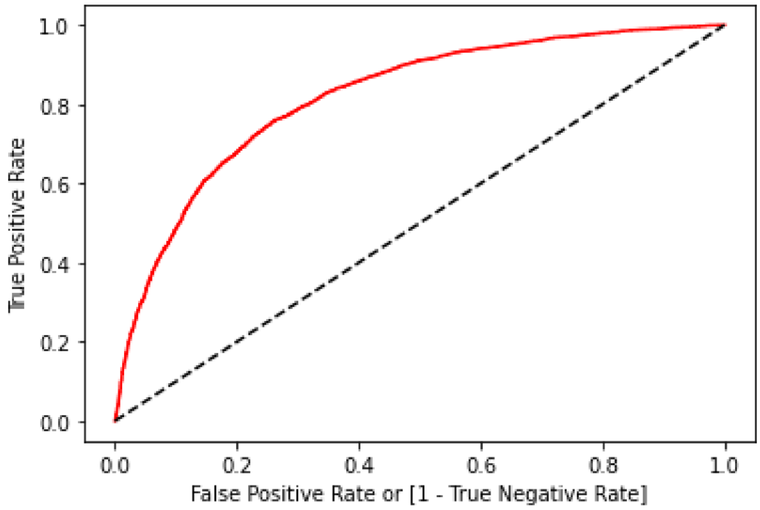
In [57]:

```
fpr, tpr, thresholds = roc_curve(under_y, log_model.predict_proba (under_x)[:,1])

auc = roc_auc_score(y_untest, y_pred_untest)
print(auc)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red', label='logit model ( area  = %0.2f)'%auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
0.7682531455533315
```

Out[57]:

```
Text(0, 0.5, 'True Positive Rate')
```



Oversampling - SMOTE

In [58]:

```
!pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\users\rooba\anaconda3\lib\site
-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\rooba\anaconda3
\lib\site-packages (from imblearn) (0.8.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\rooba\anaconda3\lib
\site-packages (from imbalanced-learn->imblearn) (1.20.1)
Requirement already satisfied: scikit-learn>=0.24 in c:\users\rooba\anaconda
3\lib\site-packages (from imbalanced-learn->imblearn) (0.24.1)
Requirement already satisfied: scipy>=0.19.1 in c:\users\rooba\anaconda3\lib
\site-packages (from imbalanced-learn->imblearn) (1.6.2)
Requirement already satisfied: joblib>=0.11 in c:\users\rooba\anaconda3\lib
\site-packages (from imbalanced-learn->imblearn) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\rooba\anacon
da3\lib\site-packages (from scikit-learn>=0.24->imbalanced-learn->imblearn)
(2.1.0)
```

Data preparation

In [60]:

```
from imblearn.over_sampling import SMOTE
```

In [61]:

```
smote=SMOTE(sampling_strategy='minority')
```

In [62]:

```
x_sm,y_sm=smote.fit_resample(x,y)
```

In [63]:

```
x_sm.shape,y_sm.shape
```

Out[63]:

```
((79844, 16), (79844,))
```

Model building|training|testing|evaluation for oversampled data

In [64]:

```
x_train_sm,x_test_sm,y_train_sm,y_test_sm=train_test_split(x_sm,y_sm,test_size=0.2,stratify
```

In [65]:

```
smote_model=LogisticRegression()
smote_model.fit(x_train_sm,y_train_sm)
```

Out[65]:

```
LogisticRegression()
```

In [66]:

```
y_pred_sm=smote_model.predict(x_test_sm)
```

In [67]:

```
print(classification_report(y_test_sm,y_pred_sm))
```

```
              precision    recall  f1-score   support

           0       0.78      0.83      0.81      7985
           1       0.82      0.76      0.79      7984

    accuracy                           0.80     15969
   macro avg       0.80      0.80      0.80     15969
weighted avg       0.80      0.80      0.80     15969
```

In [68]:

```
fpr, tpr, thresholds = roc_curve(y_sm, log_model.predict_proba (x_sm)[:,1])

auc = roc_auc_score(y_test_sm, y_pred_sm)
print(auc)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red', label='logit model ( area  = %0.2f)'%auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
0.799296471778874
```

Out[68]:

```
Text(0, 0.5, 'True Positive Rate')
```