In [1]:

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

In [9]:

```
bank=pd.read_csv('bank_data.csv')
bank
```

Out[9]:

	age	default	balance	housing	loan	duration	campaign	pdays	previous	poutfailure
0	58	0	2143	1	0	261	1	-1	0	0
1	44	0	29	1	0	151	1	-1	0	0
2	33	0	2	1	1	76	1	-1	0	0
3	47	0	1506	1	0	92	1	-1	0	0
4	33	0	1	0	0	198	1	-1	0	0
...	...	...	...	...	...	...	...	...	...	...
45206	51	0	825	0	0	977	3	-1	0	0
45207	71	0	1729	0	0	456	2	-1	0	0
45208	72	0	5715	0	0	1127	5	184	3	0
45209	57	0	668	0	0	508	4	-1	0	0
45210	37	0	2971	0	0	361	2	188	11	0

45211 rows × 32 columns

In [10]:

```
bank.shape
```

Out[10]:

```
(45211, 32)
```

In [11]:

```
bank.isna().sum()
```

Out[11]:

```
age                0
default            0
balance            0
housing            0
loan               0
duration           0
campaign           0
pdays              0
previous           0
poutfailure        0
poutother          0
poutsuccess        0
poutunknown        0
con_cellular       0
con_telephone      0
con_unknown        0
divorced           0
married            0
single             0
joadmin.           0
joblue.collar      0
joentrepreneur     0
johousemaid        0
jomanagement       0
joretired          0
joself.employed    0
joservices         0
jostudent          0
jotechnician       0
jounemployed       0
jounknown          0
y                  0
dtype: int64
```

In [13]:

```
bank_new = bank[['age','joadmin.','married','education','default','balance','housing','loan
bank_new
```

Out[13]:

	age	joadmin.	married	default	balance	housing	loan	con_telephone	duration	camp
0	58	0	1	0	2143	1	0	0	261	
1	44	0	0	0	29	1	0	0	151	
2	33	0	1	0	2	1	1	0	76	
3	47	0	1	0	1506	1	0	0	92	
4	33	0	0	0	1	0	0	0	198	
...	...	...	...	...	...	...	...	...	...	
45206	51	0	1	0	825	0	0	0	977	
45207	71	0	0	0	1729	0	0	0	456	
45208	72	0	1	0	5715	0	0	0	1127	
45209	57	0	1	0	668	0	0	1	508	
45210	37	0	1	0	2971	0	0	0	361	

45211 rows × 14 columns

In [14]:

```
# initial analysis
bank_new.shape
```

Out[14]:

```
(45211, 14)
```

In [15]:

```
bank_new.isna().sum()
```

Out[15]:

```
age              0
joadmin.         0
married          0
default          0
balance          0
housing          0
loan             0
con_telephone    0
duration         0
campaign         0
pdays            0
previous         0
poutsuccess      0
y                0
dtype: int64
```

In [16]:

```
bank_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   age            45211 non-null  int64
 1   joadmin.       45211 non-null  int64
 2   married        45211 non-null  int64
 3   default        45211 non-null  int64
 4   balance        45211 non-null  int64
 5   housing        45211 non-null  int64
 6   loan           45211 non-null  int64
 7   con_telephone  45211 non-null  int64
 8   duration       45211 non-null  int64
 9   campaign       45211 non-null  int64
 10  pdays          45211 non-null  int64
 11  previous       45211 non-null  int64
 12  poutsuccess    45211 non-null  int64
 13  y              45211 non-null  int64
dtypes: int64(14)
memory usage: 4.8 MB
```

In [17]:

```
bank_new.dtypes
```

Out[17]:

```
age              int64
joadmin.         int64
married          int64
default          int64
balance          int64
housing          int64
loan             int64
con_telephone    int64
duration         int64
campaign         int64
pdays            int64
previous         int64
poutsuccess      int64
y                int64
dtype: object
```

In [18]:

```
bank_new.describe()
```

Out[18]:

	age	joadmin.	married	default	balance	housing	
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45
mean	40.936210	0.114375	0.601933	0.018027	1362.272058	0.555838	
std	10.618762	0.318269	0.489505	0.133049	3044.765829	0.496878	
min	18.000000	0.000000	0.000000	0.000000	-8019.000000	0.000000	
25%	33.000000	0.000000	0.000000	0.000000	72.000000	0.000000	
50%	39.000000	0.000000	1.000000	0.000000	448.000000	1.000000	
75%	48.000000	0.000000	1.000000	0.000000	1428.000000	1.000000	
max	95.000000	1.000000	1.000000	1.000000	102127.000000	1.000000	

Model Building

In [19]:

```
X = bank_new.drop(labels = 'y', axis=1)
y= bank_new[['y']]
```

In [20]:

```
X
```

Out[20]:

	age	joadmin.	married	default	balance	housing	loan	con_telephone	duration	camp
0	58	0	1	0	2143	1	0	0	261	
1	44	0	0	0	29	1	0	0	151	
2	33	0	1	0	2	1	1	0	76	
3	47	0	1	0	1506	1	0	0	92	
4	33	0	0	0	1	0	0	0	198	
...	...	...	...	...	...	...	...	...	...	
45206	51	0	1	0	825	0	0	0	977	
45207	71	0	0	0	1729	0	0	0	456	
45208	72	0	1	0	5715	0	0	0	1127	
45209	57	0	1	0	668	0	0	1	508	
45210	37	0	1	0	2971	0	0	0	361	

45211 rows × 13 columns

In [21]:

```
y
```

Out[21]:

	y
0	0
1	0
2	0
3	0
4	0
...	...
45206	1
45207	1
45208	1
45209	0
45210	0

45211 rows × 1 columns

Train-Test split

In [22]:

```
from sklearn.model_selection import train_test_split
X_train,X_test, y_train,y_test = train_test_split(X,y, test_size=0.20, random_state=12)
```

In [23]:

```
X_train.shape , y_train.shape
```

Out[23]:

```
((36168, 13), (36168, 1))
```

In [25]:

```
X_test.shape , y_test.shape
```

Out[25]:

```
((9043, 13), (9043, 1))
```

Model Training

In [28]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [30]:

```
from sklearn.linear_model import LogisticRegression
logic_model = LogisticRegression()
logic_model.fit(X_train,y_train)
```

Out[30]:

```
LogisticRegression()
```

Model Testing

In [31]:

```
y_train_pred = logic_model.predict(X_train)
y_train_pred
```

Out[31]:

```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [32]:

```
y_test_pred = logic_model.predict(X_test)
y_test_pred
```

Out[32]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

Model Evaluation

In [33]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score,recall_score,cl
```

Training data

In [34]:

```
accuracy_score(y_train, y_train_pred)
```

Out[34]:

```
0.8923634151736342
```

In [35]:

```
confusion_matrix(y_train, y_train_pred)
```

Out[35]:

```
array([[31216,   713],
       [ 3180,  1059]], dtype=int64)
```

In [36]:

```
precision_score(y_train, y_train_pred)
```

Out[36]:

```
0.5976297968397292
```

In [37]:

```
recall_score(y_train, y_train_pred)
```

Out[37]:

```
0.24982307147912244
```

In [38]:

```
print(classification_report(y_train,y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     31929
           1       0.60      0.25      0.35      4239

    accuracy                           0.89     36168
   macro avg       0.75      0.61      0.65     36168
weighted avg       0.87      0.89      0.87     36168
```

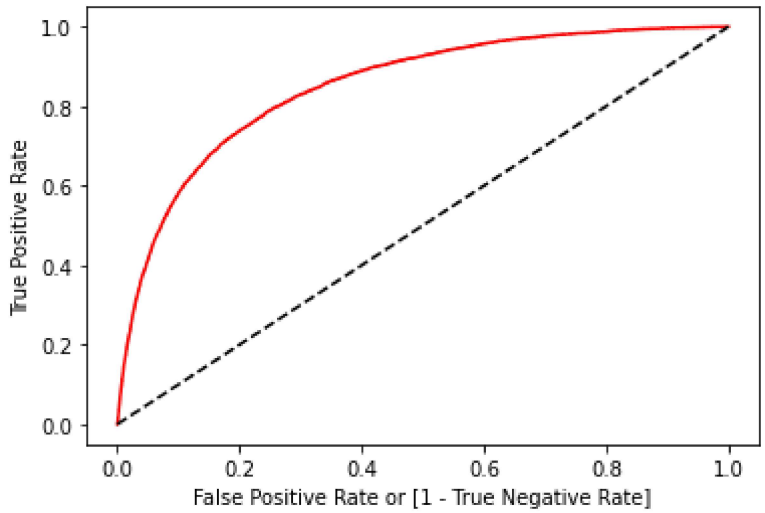
In [39]:

```
from sklearn.metrics import roc_curve,roc_auc_score
fpr, tpr, thresholds = roc_curve(y, logic_model.predict_proba (X)[:,1])

auc = roc_auc_score(y_train,y_train_pred)
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')
```

```
0.6137461375122444
```

Out[39]:

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



Test data

In [40]:

```
accuracy_score(y_test, y_test_pred)
```

Out[40]:

```
0.8932876257879022
```

In [41]:

```
confusion_matrix(y_test, y_test_pred)
```

Out[41]:

```
array([[7811,  182],
       [ 783,  267]], dtype=int64)
```

In [42]:

```
precision_score(y_test, y_test_pred)
```

Out[42]:

```
0.5946547884187082
```

In [43]:

```
recall_score(y_test, y_test_pred)
```

Out[43]:

```
0.2542857142857143
```

In [44]:

```
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      7993
           1       0.59      0.25      0.36      1050

    accuracy                           0.89      9043
   macro avg       0.75      0.62      0.65      9043
weighted avg       0.87      0.89      0.87      9043
```

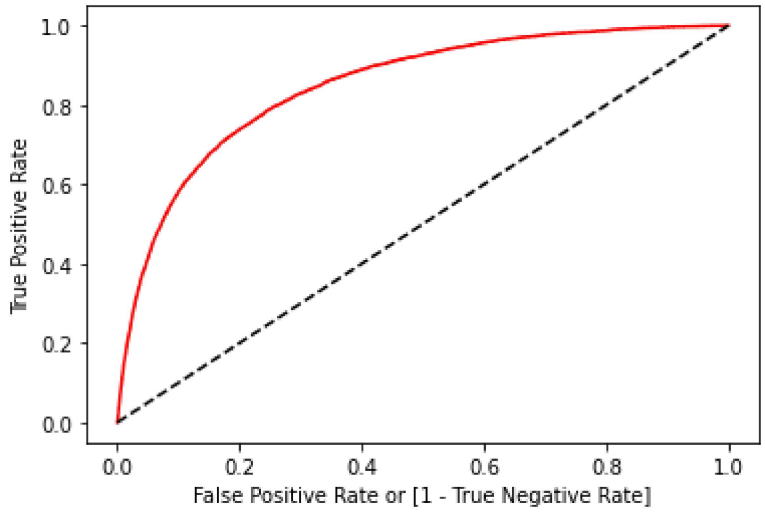
In [45]:

```
from sklearn.metrics import roc_curve,roc_auc_score
fpr, tpr, thresholds = roc_curve(y, logic_model.predict_proba (X)[:,1])

auc = roc_auc_score(y_test,y_test_pred)
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')
```

```
0.6157578953012457
```

Out[45]:

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



Model Deployement

In [46]:

```
from pickle import dump
```

In [49]:

```
dump(logic_model,open('logic_model.pkl','wb'))
```

In [50]:

```
from pickle import load
```

In [51]:

```
logistic_model_pickle = load(open('logic_model.pkl','rb'))
```

In [53]:

```
pickle_pred = logistic_model_pickle.predict(X_test)
```

In [ ]:

```
```

Decision Tree Performance

In [54]:

```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train,y_train)
```

Out[54]:

```
DecisionTreeClassifier()
```

Model testing

In [56]:

```
y_train_pred =dt_model.predict(X_train) #training data
```

In [57]:

```
y_test_pred = dt_model.predict(X_test) #test data
```

Model Evaluation

In [58]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score,recall_score,cl
```

In [59]:

```
accuracy_score(y_train,y_train_pred)
```

Out[59]:

```
1.0
```

In [60]:

```
confusion_matrix(y_train,y_train_pred)
```

Out[60]:

```
array([[31929,     0],
       [    0,  4239]], dtype=int64)
```

In [61]:

```
precision_score(y_train,y_train_pred)
```

Out[61]:

```
1.0
```

In [62]:

```
recall_score(y_train,y_train_pred)
```

Out[62]:

```
1.0
```

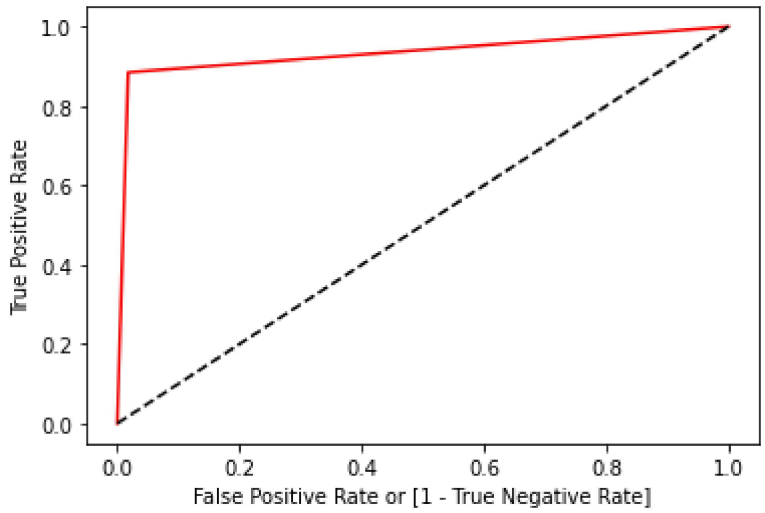
In [63]:

```
from sklearn.metrics import roc_curve,roc_auc_score
fpr, tpr, thresholds = roc_curve(y, dt_model.predict_proba (X)[:,1])

auc = roc_auc_score(y_train,y_train_pred)
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')
```

```
1.0
```

Out[63]:

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



In [ ]:

```

```

In [64]:

```
accuracy_score(y_test, y_test_pred)
```

Out[64]:

```
0.8544730730952118
```

In [65]:

```
confusion_matrix(y_test, y_test_pred)
```

Out[65]:

```
array([[7287,  706],
       [ 610,  440]], dtype=int64)
```

In [66]:

```
precision_score(y_test, y_test_pred)
```

Out[66]:

```
0.38394415357766143
```

In [67]:

```
recall_score(y_test, y_test_pred)
```

Out[67]:

```
0.41904761904761906
```

In [68]:

```
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.91      0.92      7993
           1       0.38      0.42      0.40      1050

    accuracy                           0.85      9043
   macro avg       0.65      0.67      0.66      9043
weighted avg       0.86      0.85      0.86      9043
```

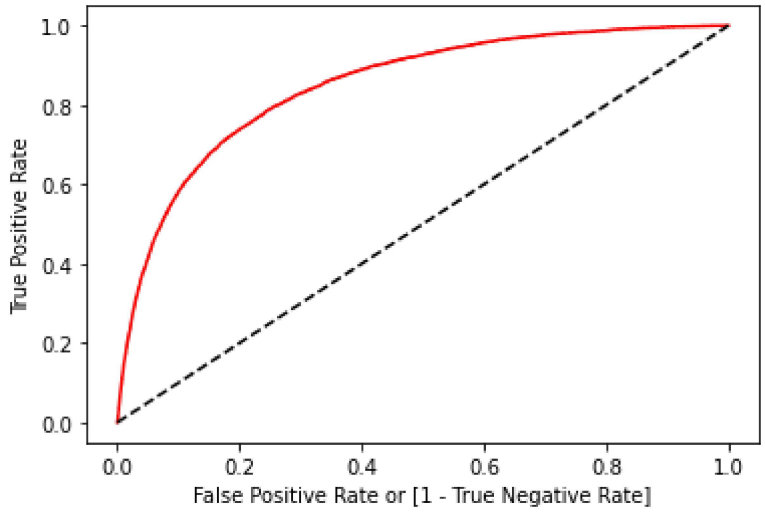
In [69]:

```
from sklearn.metrics import roc_curve,roc_auc_score
fpr, tpr, thresholds = roc_curve(y, logic_model.predict_proba (X)[:,1])

auc = roc_auc_score(y_test,y_test_pred)
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')
```

```
0.6653601663360202
```

Out[69]:

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



In [ ]: