In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

In [3]:

```
bank = pd.read_csv('/Users/acer/Sandesh Pal/Data Science Assgn/LOgistic Regression/bank-full.csv',sep=';'
```

In [4]:

```
#Checking all the columns
bank.head()
```

Out[4]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown

In [5]:

```
bank.y.replace(('yes','no'),(1,0),inplace=True)
```

In [6]:

```
bank.default.replace(('yes','no'),(1,0),inplace=True)
```

In [7]:

```
bank.housing.replace(('yes','no'),(1,0),inplace=True)
```

In [8]:

```
bank.loan.replace(('yes','no'),(1,0),inplace=True)
```

In [9]:

```
bank.head(3)
```

Out[9]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	58	management	married	tertiary	0	2143	1	0	unknown	5	may	261	1	-1	0	unknown
1	44	technician	single	secondary	0	29	1	0	unknown	5	may	151	1	-1	0	unknown
2	33	entrepreneur	married	secondary	0	2	1	1	unknown	5	may	76	1	-1	0	unknown

In [10]:

```
#Checking for na values
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  int64
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  int64
 7   loan       45211 non-null  int64
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  int64
dtypes: int64(11), object(6)
memory usage: 5.9+ MB
```

In [11]:

```
bank = pd.get_dummies(bank)
```

In [12]:

```
bank.head(3)
```

Out[12]:

	age	default	balance	housing	loan	day	duration	campaign	pdays	previous	...	month_jun	month_mar	month_may	month_nov	month_c
0	58	0	2143	1	0	5	261	1	-1	0	...	0	0	1	0	
1	44	0	29	1	0	5	151	1	-1	0	...	0	0	1	0	
2	33	0	2	1	1	5	76	1	-1	0	...	0	0	1	0	

3 rows × 49 columns

In [13]:

```
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 49 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   age                  45211 non-null  int64
 1   default              45211 non-null  int64
 2   balance              45211 non-null  int64
 3   housing              45211 non-null  int64
 4   loan                 45211 non-null  int64
 5   day                  45211 non-null  int64
 6   duration             45211 non-null  int64
 7   campaign             45211 non-null  int64
 8   pdays                45211 non-null  int64
 9   previous             45211 non-null  int64
 10  y                    45211 non-null  int64
 11  job_admin.           45211 non-null  uint8
 12  job_blue-collar      45211 non-null  uint8
 13  job_entrepreneur     45211 non-null  uint8
 14  job_housemaid        45211 non-null  uint8
 15  job_management       45211 non-null  uint8
 16  job_retired          45211 non-null  uint8
 17  job_self-employed    45211 non-null  uint8
 18  job_services         45211 non-null  uint8
 19  job_student          45211 non-null  uint8
 20  job_technician       45211 non-null  uint8
 21  job_unemployed       45211 non-null  uint8
 22  job_unknown          45211 non-null  uint8
 23  marital_divorced     45211 non-null  uint8
 24  marital_married      45211 non-null  uint8
 25  marital_single       45211 non-null  uint8
 26  education_primary    45211 non-null  uint8
 27  education_secondary  45211 non-null  uint8
 28  education_tertiary   45211 non-null  uint8
 29  education_unknown    45211 non-null  uint8
 30  contact_cellular     45211 non-null  uint8
 31  contact_telephone    45211 non-null  uint8
 32  contact_unknown      45211 non-null  uint8
 33  month_apr            45211 non-null  uint8
 34  month_aug            45211 non-null  uint8
 35  month_dec            45211 non-null  uint8
 36  month_feb            45211 non-null  uint8
 37  month_jan            45211 non-null  uint8
 38  month_jul            45211 non-null  uint8
 39  month_jun            45211 non-null  uint8
 40  month_mar            45211 non-null  uint8
 41  month_may            45211 non-null  uint8
 42  month_nov            45211 non-null  uint8
 43  month_oct            45211 non-null  uint8
 44  month_sep            45211 non-null  uint8
 45  poutcome_failure     45211 non-null  uint8
 46  poutcome_other       45211 non-null  uint8
 47  poutcome_success     45211 non-null  uint8
 48  poutcome_unknown     45211 non-null  uint8
dtypes: int64(11), uint8(38)
memory usage: 5.4 MB
```

In [14]:

```
bank[bank.duplicated()]
```

Out[14]:

age	default	balance	housing	loan	day	duration	campaign	pdays	previous	...	month_jun	month_mar	month_may	month_nov	month_oc

0 rows × 49 columns

In [15]:

```
#Dividing the dataset into X and Y variables
X = bank.loc[:,bank.columns!='y']
Y = np.ravel(bank.loc[:,bank.columns=='y'])
```

In [16]:

```
X
```

Out[16]:

	age	default	balance	housing	loan	day	duration	campaign	pdays	previous	...	month_jun	month_mar	month_may	month_nov	mor
0	58	0	2143	1	0	5	261	1	-1	0	...	0	0	1	0	
1	44	0	29	1	0	5	151	1	-1	0	...	0	0	1	0	
2	33	0	2	1	1	5	76	1	-1	0	...	0	0	1	0	
3	47	0	1506	1	0	5	92	1	-1	0	...	0	0	1	0	
4	33	0	1	0	0	5	198	1	-1	0	...	0	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	0	825	0	0	17	977	3	-1	0	...	0	0	0	1	
45207	71	0	1729	0	0	17	456	2	-1	0	...	0	0	0	1	
45208	72	0	5715	0	0	17	1127	5	184	3	...	0	0	0	1	
45209	57	0	668	0	0	17	508	4	-1	0	...	0	0	0	1	
45210	37	0	2971	0	0	17	361	2	188	11	...	0	0	0	1	

45211 rows × 48 columns

In [17]:

```
Y
```

Out[17]:

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

In [18]:

```
#Building the logistic regression modellalit mehendiratta
model = LogisticRegression()
model.fit(X,Y)
```

```
C:\Users\acer\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[18]:

```
LogisticRegression()
```

In [19]:

```
#Predict for X dataset
y_pred = model.predict(X)
```

In [20]:

```
y_pred_df= pd.DataFrame({'actual': Y,
                         'predicted_prob': y_pred})
```

In [21]:

```
y_pred_df
```

Out[21]:

	actual	predicted_prob
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
...	...	...
45206	1	1
45207	1	0
45208	1	1
45209	0	0
45210	0	0

45211 rows × 2 columns

In [22]:

```
# Confusion Matrix for the model accuracy
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(Y,y_pred)
print (confusion_matrix)
```

```
[[39165   757]
 [ 4146  1143]]
```

In [23]:

```
#Classification report
from sklearn.metrics import classification_report
print(classification_report(Y,y_pred))
```

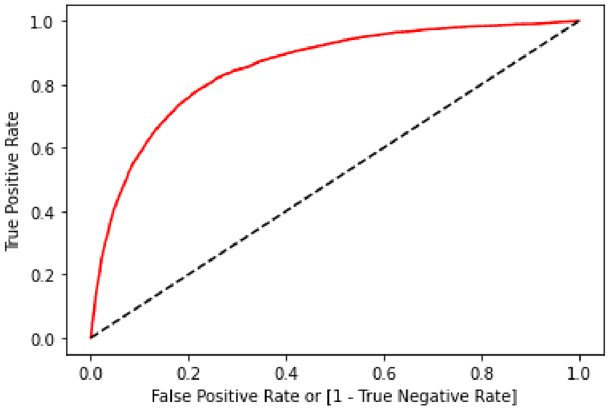
```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94     39922
           1       0.60      0.22      0.32      5289

    accuracy                           0.89     45211
   macro avg       0.75      0.60      0.63     45211
weighted avg       0.87      0.89      0.87     45211
```

In [24]:

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

fpr, tpr, thresholds = roc_curve(Y, model.predict_proba (X)[:,1])

auc = roc_auc_score(Y, y_pred)

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')
```

Out[24]:

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



In [25]:

```
auc
```

Out[25]:

```
0.5985734647110931
```

In [ ]: