In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
dataset=pd.read_csv(r"C:\Users\kotha\Downloads\bank.csv")
```

In [3]:

```
dataset.head()
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	d
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	

In [4]:

```
dataset.isnull().any()
```

Out[4]:

```
age          False
job          False
marital      False
education    False
default      False
balance      False
housing      False
loan         False
contact      False
day          False
month        False
duration     False
campaign     False
pdays        False
previous     False
poutcome     False
deposit      False
dtype: bool
```

In [5]:

```
dataset.describe()
```

Out[5]:

	age	balance	day	duration	campaign	pdays
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000

In [6]:

```
dataset.corr()
```

Out[6]:

	age	balance	day	duration	campaign	pdays	previous
age	1.000000	0.112300	-0.000762	0.000189	-0.005278	0.002774	0.020169
balance	0.112300	1.000000	0.010467	0.022436	-0.013894	0.017411	0.030805
day	-0.000762	0.010467	1.000000	-0.018511	0.137007	-0.077232	-0.058981
duration	0.000189	0.022436	-0.018511	1.000000	-0.041557	-0.027392	-0.026716
campaign	-0.005278	-0.013894	0.137007	-0.041557	1.000000	-0.102726	-0.049699
pdays	0.002774	0.017411	-0.077232	-0.027392	-0.102726	1.000000	0.507272
previous	0.020169	0.030805	-0.058981	-0.026716	-0.049699	0.507272	1.000000

In [7]:

```
x= dataset.iloc[:,0:16].values
x
```

Out[7]:

```
array([[59, 'admin.', 'married', ..., -1, 0, 'unknown'],
       [56, 'admin.', 'married', ..., -1, 0, 'unknown'],
       [41, 'technician', 'married', ..., -1, 0, 'unknown'],
       ...,
       [32, 'technician', 'single', ..., -1, 0, 'unknown'],
       [43, 'technician', 'married', ..., 172, 5, 'failure'],
       [34, 'technician', 'married', ..., -1, 0, 'unknown']], dtype=object)
```

In [8]:

```
x.shape
```

Out[8]:

```
(11162, 16)
```

In [9]:

```
y= dataset.iloc[:,16:17].values
y
```

Out[9]:

```
array([['yes'],
       ['yes'],
       ['yes'],
       ...,
       ['no'],
       ['no'],
       ['no']], dtype=object)
```

In [10]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
```

In [11]:

```
ct= ColumnTransformer([("oh",OneHotEncoder(),[1,2,3,4,6,7,8,10,15])],remainder="passthrough
x=ct.fit_transform(x)
x
```

Out[11]:

```
array([[1.0, 0.0, 0.0, ..., 1, -1, 0],
       [1.0, 0.0, 0.0, ..., 1, -1, 0],
       [0.0, 0.0, 0.0, ..., 1, -1, 0],
       ...,
       [0.0, 0.0, 0.0, ..., 2, -1, 0],
       [0.0, 0.0, 0.0, ..., 2, 172, 5],
       [0.0, 0.0, 0.0, ..., 1, -1, 0]], dtype=object)
```

In [12]:

```
ct1= ColumnTransformer([("oh",OneHotEncoder(),[0])],remainder="passthrough")
y=ct1.fit_transform(y)
y
```

Out[12]:

```
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       ...,
       [1., 0.],
       [1., 0.],
       [1., 0.]])
```

In [13]:

```
y=y[:,1:]
y
```

Out[13]:

```
array([[1.],
       [1.],
       [1.],
       ...,
       [0.],
       [0.],
       [0.]])
```

In [14]:

```
import joblib
joblib.dump(ct,"column")
```

Out[14]:

```
['column']
```

In [15]:

```
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,random_state=0)
```

In [16]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

In [17]:

```
x_train.shape
```

Out[17]:

```
(8929, 51)
```

In [18]:

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

```
C:\Users\kotha\anaconda3\lib\site-packages\sklearn\utils\validation.py:760:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
```

Out[18]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [19]:

```
x_train[0]
```

Out[19]:

```
array([-0.37172192, -0.45760242, -0.17658264, -0.1589378 ,  1.8452729 ,
       -0.27363591, -0.1931085 , -0.30435733, -0.17992689, -0.44212513,
       -0.18025832, -0.08293772, -0.36485864,  0.86351031, -0.66975544,
       -0.39346051,  1.01614555, -0.69993206, -0.21741017,  0.12621015,
       -0.12621015, -1.05825117,  1.05825117,  0.39003722, -0.39003722,
       -1.60273048, -0.27292594,  1.93439945, -0.30193478, -0.39913565,
       -0.09686471, -0.27410848, -0.17488949, -0.39459853, -0.34910544,
       -0.15930776,  1.70870313, -0.30215554, -0.1931085 , -0.16761389,
       -0.35031555, -0.22657536, -0.32939951,  0.58661881,  0.64941897,
       -0.44565283, -0.90504798, -0.2483547 , -0.56394809, -0.48428507,
       -0.35888264])
```

In [20]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [21]:

```
rf= RandomForestClassifier(n_estimators=100,criterion="entropy",random_state=0)
rf.fit(x_train,y_train)
```

```
C:\Users\kotha\anaconda3\lib\site-packages\ipykernel_launcher.py:2: DataConv
ersionWarning: A column-vector y was passed when a 1d array was expected. Pl
ease change the shape of y to (n_samples,), for example using ravel().
```

Out[21]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='au
to',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=0, verbose
=0,
                       warm_start=False)
```

In [22]:

```
y_pred=rf.predict(x_test)
y_pred
```

Out[22]:

```
array([0., 1., 0., ..., 1., 0., 0.])
```

In [23]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[23]:

```
0.8477384684281236
```

In [24]:

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
cm
```

Out[24]:

```
array([[1005,  200],
       [ 140,  888]], dtype=int64)
```

In [25]:

```
import sklearn.metrics as metrics
fpr,tpr,threshold=metrics.roc_curve(y_test,y_pred)
roc_auc=metrics.auc(fpr,tpr)
```

In [26]:

```
roc_auc
```

Out[26]:

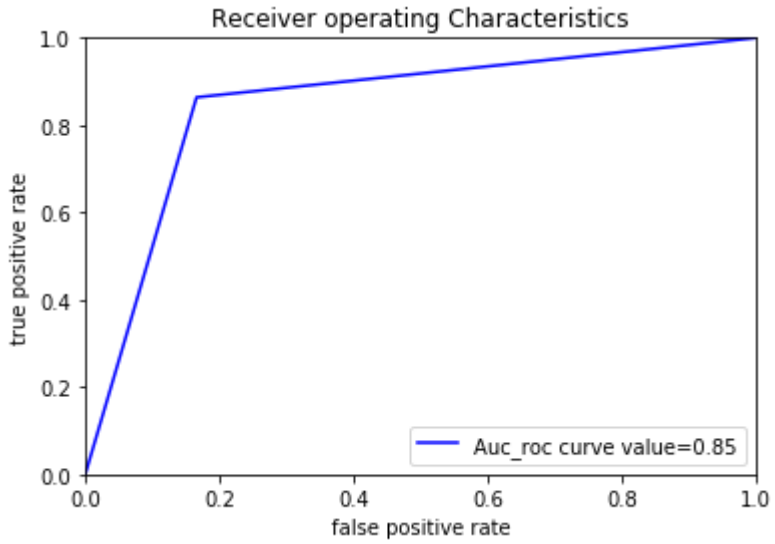
```
0.8489190629187723
```

In [27]:

```
plt.title("Receiver operating Characteristics")
plt.plot(fpr,tpr,color="blue",label='Auc_roc curve value=%0.2f'%roc_auc)
plt.legend()
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel("true positive rate")
plt.xlabel("false positive rate")
```

Out[27]:

```
Text(0.5, 0, 'false positive rate')
```



In [28]:

```
new=joblib.load('column')
```

In [29]:

```
p=new.transform([[59,'admin.','married','secondary','no',2343,'yes','no','unknown','5','may
```

In [30]:

```
p
```

Out[30]:

```
array([[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
        1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0,
        0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
        0.0, 0.0, 0.0, 0.0, 1.0, 59, 2343, '5', 1042, 1, -1, 0]],
      dtype=object)
```

In [31]:

```
rf.predict(p)
```

Out[31]:

```
array([1.])
```

In [32]:

```
import pickle
pickle.dump(rf,open('deposit.pkl','wb'))
```

In [ ]:

In [ ]: