

In [1]:

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

In [2]:

```
dataset=pd.read_csv('Churn_Modelling1.csv')
```

In [3]:

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

Out[3]:

RowNumber	False
CustomerId	False
Surname	False
CreditScore	False
Geography	True
Gender	True
Age	True
Tenure	False
Balance	True
NumOfProducts	False
HasCrCard	False
IsActiveMember	False
EstimatedSalary	False
Exited	False
dtype:	bool

In [4]:

```
dataset[dataset ['Age'].isnull()].index.tolist()
```

Out[4]:

```
[4, 28, 43, 59]
```

In [5]:

```
dataset['Age'].fillna(dataset['Age'].mean(), inplace=True)
```

In [6]:

```
dataset[dataset ['Gender'].isnull()].index.tolist()
```

Out[6]:

```
[6, 21, 32]
```

In [8]:

```
dataset['Gender'].fillna(dataset['Gender'].mode(), inplace=True)
```

```
In [14]: dataset['Balance'].fillna(dataset['Balance'].mean(),inplace=True)
```

```
In [13]: dataset[dataset['Balance'].isnull()].index.tolist()
```

```
Out[13]: []
```

```
In [15]: dataset['Gender'] = dataset['Gender'].fillna(dataset['Gender'].mode()[0])
```

```
In [16]: dataset['Geography'] = dataset['Geography'].fillna(dataset['Geography'].mode()[0])
```

```
In [18]: dataset.isnull().any()
```

```
Out[18]: RowNumber      False
CustomerId    False
Surname        False
CreditScore    False
Geography      False
Gender         False
Age           False
Tenure        False
Balance        False
NumOfProducts False
HasCrCard     False
IsActiveMember False
EstimatedSalary False
Exited        False
dtype: bool
```

```
In [19]: dataset['Gender'].fillna(dataset['Gender'].mode()[0],inplace=True)
dataset['Geography'].fillna(dataset['Geography'].mode()[0],inplace=True)
```

```
In [20]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset['Gender'] = le.fit_transform(dataset['Gender'])
dataset['Geography'] = le.fit_transform(dataset['Geography'])
```

In [22]: `dataset.corr()`

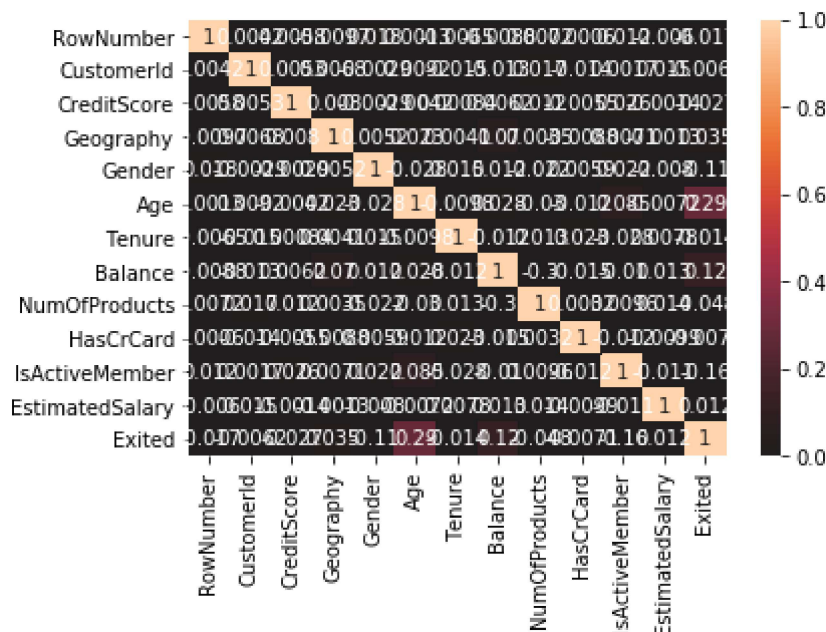
Out[22]:

Number	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProd
000000	0.004202	0.005840	-0.009734	0.017850	0.001279	-0.006495	-0.008830	0.001
004202	1.000000	0.005308	0.006779	-0.002901	0.009184	-0.014883	-0.012525	0.016
005840	0.005308	1.000000	0.008034	-0.002887	-0.004154	0.000842	0.006215	0.012
009734	0.006779	0.008034	1.000000	0.005178	0.022713	0.004075	0.069609	0.003
017850	-0.002901	-0.002887	0.005178	1.000000	-0.027595	0.014941	0.011716	-0.027
001279	0.009184	-0.004154	0.022713	-0.027595	1.000000	-0.009790	0.028282	-0.030
006495	-0.014883	0.000842	0.004075	0.014941	-0.009790	1.000000	-0.012168	0.013
008830	-0.012525	0.006215	0.069609	0.011716	0.028282	-0.012168	1.000000	-0.304
007246	0.016972	0.012238	0.003472	-0.021697	-0.030420	0.013444	-0.304141	1.000
000599	-0.014025	-0.005458	-0.008757	0.005896	-0.011698	0.022583	-0.014858	0.003
012044	0.001665	0.025651	0.007098	0.022338	0.085395	-0.028362	-0.010136	0.009
005988	0.015271	-0.001384	-0.001339	-0.007978	-0.007239	0.007784	0.012695	0.014
016571	-0.006248	-0.027094	0.035227	-0.106616	0.285267	-0.014001	0.118609	-0.047



In [23]: `import seaborn as sns`
`sns.heatmap(dataset.corr(),annot=True,vmin=0,vmax=1,center=0)`

Out[23]: `<matplotlib.axes._subplots.AxesSubplot at 0x1f31843b2c8>`



```
In [24]: y = dataset.iloc[:, -1:].values
y
```

```
Out[24]: array([[1],
                [0],
                [1],
                ...,
                [1],
                [1],
                [0]], dtype=int64)
```

```
In [25]: x = dataset.iloc[:, 0:-1].values
x
```

```
Out[25]: array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
                [2, 15647311, 'Hill', ..., 0, 1, 112542.58],
                [3, 15619304, 'Onio', ..., 1, 0, 113931.57],
                ...,
                [9998, 15584532, 'Liu', ..., 0, 1, 42085.58],
                [9999, 15682355, 'Sabbatini', ..., 1, 0, 92888.52],
                [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]], dtype=object)
```

```
In [26]: x = np.delete(x, 0, axis=1)
x
```

```
Out[26]: array([[15634602, 'Hargrave', 619, ..., 1, 1, 101348.88],
                [15647311, 'Hill', 608, ..., 0, 1, 112542.58],
                [15619304, 'Onio', 502, ..., 1, 0, 113931.57],
                ...,
                [15584532, 'Liu', 709, ..., 0, 1, 42085.58],
                [15682355, 'Sabbatini', 772, ..., 1, 0, 92888.52],
                [15628319, 'Walker', 792, ..., 1, 0, 38190.78]], dtype=object)
```

```
In [27]: x = np.delete(x, 0, axis=1)
x = np.delete(x, 0, axis=1)
```

```
In [28]: x
```

```
Out[28]: array([[619, 0, 0, ..., 1, 1, 101348.88],
                [608, 2, 0, ..., 0, 1, 112542.58],
                [502, 0, 0, ..., 1, 0, 113931.57],
                ...,
                [709, 0, 0, ..., 0, 1, 42085.58],
                [772, 1, 1, ..., 1, 0, 92888.52],
                [792, 0, 0, ..., 1, 0, 38190.78]], dtype=object)
```

```
In [29]: from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder()
```

```
In [30]: z = oh.fit_transform(x[:,1:2]).toarray()  
z
```

```
Out[30]: array([[1., 0., 0.],  
               [0., 0., 1.],  
               [1., 0., 0.],  
               ...,  
               [1., 0., 0.],  
               [0., 1., 0.],  
               [1., 0., 0.]])
```

```
In [31]: x = np.delete(x,1,axis=1)
```

```
In [32]: x = np.concatenate((x,z),axis=1)
```

```
In [33]: z = oh.fit_transform(x[:,1:2]).toarray()  
z
```

```
Out[33]: array([[1., 0.],  
               [1., 0.],  
               [1., 0.],  
               ...,  
               [1., 0.],  
               [0., 1.],  
               [1., 0.]])
```

```
In [34]: x = np.delete(x,1,axis=1)  
x = np.concatenate((x,z),axis=1)  
x.shape
```

```
Out[34]: (10000, 13)
```

```
In [35]: 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_s
```

```
In [36]: from sklearn.preprocessing import StandardScaler  
sc = StandardScaler()  
x_train = sc.fit_transform(x_train)  
x_test = sc.fit_transform(x_test)
```

```
In [38]: from sklearn.tree import DecisionTreeClassifier  
dtc = DecisionTreeClassifier(criterion = 'entropy' )  
dtc.fit(x_train,y_train)
```

```
Out[38]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',  
                                max_depth=None, max_features=None, max_leaf_nodes=None,  
                                min_impurity_decrease=0.0, min_impurity_split=None,  
                                min_samples_leaf=1, min_samples_split=2,  
                                min_weight_fraction_leaf=0.0, presort='deprecated',  
                                random_state=None, splitter='best')
```

```
In [39]: y_pred = dtc.predict(x_test)
y_pred
```

```
Out[39]: array([1, 0, 0, ..., 0, 1, 1], dtype=int64)
```

```
In [40]: y_test
```

```
Out[40]: array([[0],
                [1],
                [0],
                ...,
                [0],
                [0],
                [0]], dtype=int64)
```

```
In [41]: from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

```
Out[41]: 0.7985
```

```
In [42]: from sklearn.metrics import confusion_matrix
```

```
In [43]: cm=confusion_matrix(y_test,y_pred)
```

```
In [45]: cm
dataset.head
```

```
Out[45]: <bound method NDFrame.head of
re Geography Gender \
0      1  15634602  Hargrave      619      0      0
1      2  15647311    Hill      608      2      0
2      3  15619304    Onio      502      0      0
3      4  15701354    Boni      699      0      0
4      5  15737888  Mitchell      850      2      0
...    ...    ...    ...    ...    ...    ...
9995   9996  15606229  Obijiaku      771      0      1
9996   9997  15569892  Johnstone      516      0      1
9997   9998  15584532    Liu      709      0      0
9998   9999  15682355  Sabbatini      772      1      1
9999  10000  15628319    Walker      792      0      0

      Age  Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0  42.000000      2      0.00           1           1           1
1  41.000000      1  83807.86           1           0           1
2  42.000000      8  159660.80           3           1           0
3  39.000000      1      0.00           2           0           0
4  38.918768      2  125510.82           1           1           1
...    ...    ...    ...    ...    ...    ...
9995  39.000000      5      0.00           2           1           0
9996  35.000000     10   57369.61           1           1           1
9997  36.000000      7      0.00           1           0           1
9998  42.000000      3   75075.31           2           1           0
9999  28.000000      4  130142.79           1           1           0

      EstimatedSalary  Exited
0      101348.88      1
1      112542.58      0
2      113931.57      1
3      93826.63      0
4      79084.10      0
...    ...    ...
9995      96270.64      0
9996     101699.77      0
9997      42085.58      1
9998      92888.52      1
9999      38190.78      0

[10000 rows x 14 columns]>
```

```
In [47]: from sklearn import tree
tree.export_graphviz(dtc)

from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

dot_data=StringIO()
export_graphviz(dtc,out_file=dot_data,
                filled=True,rounded=True,
                special_characters=True)

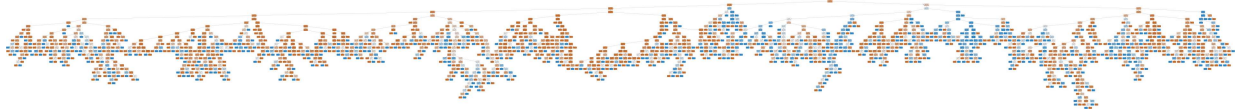
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

C:\Users\PRUDHVI\anaconda3\lib\site-packages\sklearn\externals\six.py:31: FutureWarning: The module is deprecated in version 0.21 and will be removed in version 0.23 since we've dropped support for Python 2.7. Please rely on the official version of six (<https://pypi.org/project/six/>).

"(https://pypi.org/project/six/).", FutureWarning)

dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.703834 to fit

Out[47]:



```
In [49]: #random forest on churn modelling
```

```
In [48]: from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=30,criterion='entropy',max_depth=15,max
rfc.fit(x_train,y_train)
y_pred_rfc = rfc.predict(x_test)
y_pred_rfc
```

C:\Users\PRUDHVI\anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

This is separate from the ipykernel package so we can avoid doing imports until

Out[48]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)


```
In [50]: accuracy_score(y_test,y_pred_rfc)
```

```
Out[50]: 0.8315
```

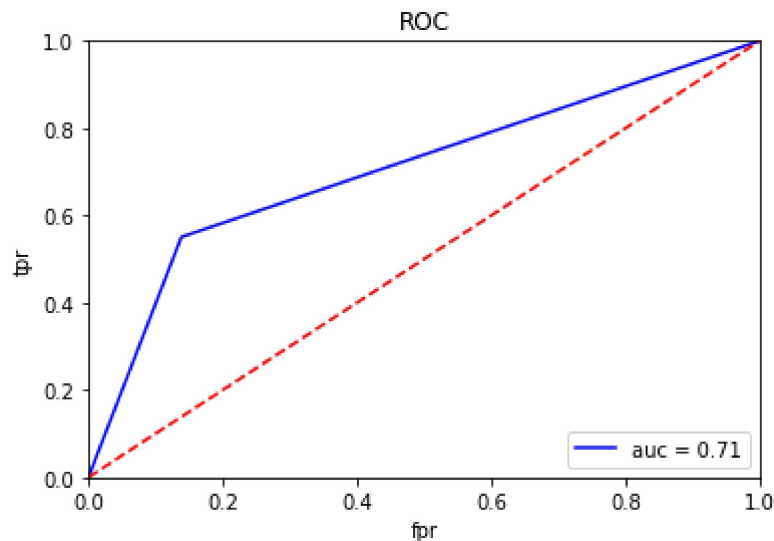
```
In [51]: confusion_matrix(y_test,y_pred_rfc)
```

```
Out[51]: array([[1589,    6],
                [ 331,   74]], dtype=int64)
```

```
In [52]: import sklearn.metrics as metrics
fpr,tpr,threshold = metrics.roc_curve(y_test,y_pred)
roc_auc = metrics.auc(fpr,tpr)

import matplotlib.pyplot as plt
plt.title("ROC")
plt.plot(fpr,tpr,'b',label = 'auc = %0.2f'%roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('fpr')
plt.ylabel('tpr')
```

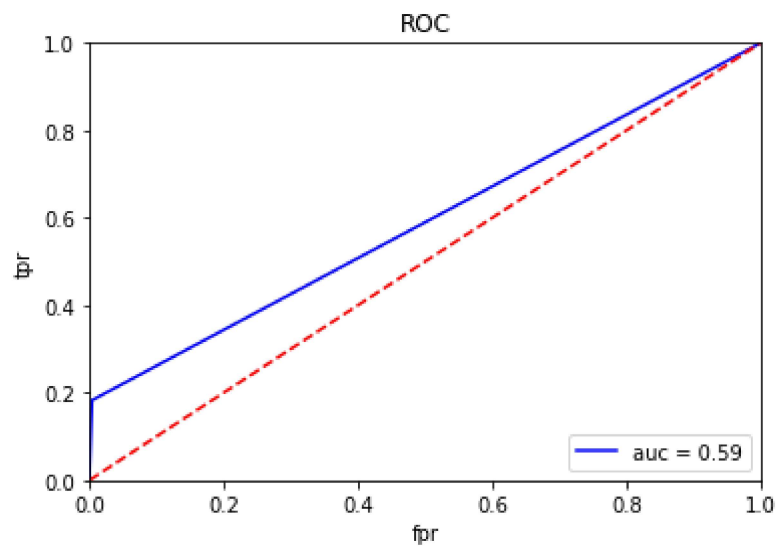
```
Out[52]: Text(0, 0.5, 'tpr')
```



```
In [53]: #auc, #roc
#area under curve
import sklearn.metrics as metrics
fpr, tpr, threshold = metrics.roc_curve(y_test, y_pred_rfc)
roc_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt
plt.title("ROC")
plt.plot(fpr, tpr, 'b', label = 'auc = %0.2f'%roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1], 'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('fpr')
plt.ylabel('tpr')
```

Out[53]: Text(0, 0.5, 'tpr')



In []:

