we are going to find the threshold for the binary classification problem

- we are going to create our own dataset

In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
# roc curve and auc score
from sklearn.datasets import make_classification
```

In [7]:
```
from sklearn.model_selection import train_test_split
X,y=make_classification(n_samples=2000,n_classes=2,weights=[1,1],random_state=1)
```

In [8]:
```
X.shape
```

Out[8]:
```
(2000, 20)
```

In [9]:
```
y
```

Out[9]:
```
array([0, 0, 0, ..., 1, 1, 0])
```

In [10]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1)
```

In [11]:
```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

In [12]:
```
## RandomForest classifier
# now applying the random forest classifier
```

In [15]:
```
from sklearn.ensemble import RandomForestClassifier
rf_model=RandomForestClassifier()
rf_model.fit(X_train,y_train)
ytrain_pred=rf_model.predict_proba(X_train)
print('RF train roc-auc: {}'.format(roc_auc_score(y_train, ytrain_pred[:,1])))
ytest_pred = rf_model.predict_proba(X_test)
print('RF test roc-auc: {}'.format(roc_auc_score(y_test, ytest_pred[:,1])))
```

```
RF train roc-auc: 0.9999999999999999
RF test roc-auc: 0.9843666666666666
```

In [19]:
```
ytrain_pred
```

Out[19]:
```
array([[0.99, 0.01],
       [0.98, 0.02],
       [0.01, 0.99],
       ...,
       [0.96, 0.04],
       [0.98, 0.02],
       [0.31, 0.69]])
```

In [20]:
```
## Logistic Regression
```

In [21]:
```
from sklearn.linear_model import LogisticRegression
log_classifier=LogisticRegression()
log_classifier.fit(X_train, y_train)
ytrain_pred = log_classifier.predict_proba(X_train)
print('Logistic train roc-auc: {}'.format(roc_auc_score(y_train, ytrain_pred[:,1])))
ytest_pred = log_classifier.predict_proba(X_test)
print('Logistic test roc-auc: {}'.format(roc_auc_score(y_test, ytest_pred[:,1])))
```

```
Logistic train roc-auc: 0.9863568922694498
Logistic test roc-auc: 0.9885777777777777
```

In [22]:
```
## Adaboost classifier
```

In [23]:
```
from sklearn.ensemble import AdaBoostClassifier
ada_classifier=AdaBoostClassifier()
ada_classifier.fit(X_train, y_train)
ytrain_pred = ada_classifier.predict_proba(X_train)
print('Adaboost train roc-auc: {}'.format(roc_auc_score(y_train, ytrain_pred[:,1])))
ytest_pred = ada_classifier.predict_proba(X_test)
print('Adaboost test roc-auc: {}'.format(roc_auc_score(y_test, ytest_pred[:,1])))
```

```
Adaboost train roc-auc: 0.9975081174960356
Adaboost test roc-auc: 0.9826111111111111
```

In [24]:
```
## KNN classifier
```

In [26]:
```
from sklearn.neighbors import KNeighborsClassifier
knn_classifier=KNeighborsClassifier()
knn_classifier.fit(X_train, y_train)
ytrain_pred = knn_classifier.predict_proba(X_train)
print('Adaboost train roc-auc: {}'.format(roc_auc_score(y_train, ytrain_pred[:,1])))
ytest_pred = knn_classifier.predict_proba(X_test)
print('Adaboost test roc-auc: {}'.format(roc_auc_score(y_test, ytest_pred[:,1])))
```

```
Adaboost train roc-auc: 0.9816700714491109
Adaboost test roc-auc: 0.9426111111111111
```

Now we will focus on selecting the best threshold for maximum accuracy

In [28]:
```
pred=[]
for model in [rf_model,log_classifier,ada_classifier,knn_classifier]:
    pred.append(pd.Series(model.predict_proba(X_test)[:,1]))
final_prediction=pd.concat(pred,axis=1).mean(axis=1)
print('Ensemble test roc-auc: {}'.format(roc_auc_score(y_test,final_prediction)))
```

```
Ensemble test roc-auc: 0.9852333333333333
```

In [29]:
```
pd.concat(pred,axis=1)
```

Out[29]:

	0	1	2	3
0	0.99	0.991861	0.559186	1.0
1	0.02	0.000008	0.463282	0.0
2	0.97	0.966929	0.538202	0.8
3	0.93	0.761539	0.509875	0.8
4	0.59	0.779443	0.490344	0.4
...	...	...	...	...
595	0.02	0.024239	0.461121	0.0
596	0.02	0.000003	0.441377	0.0
597	0.98	0.984385	0.532403	1.0
598	0.01	0.001147	0.441720	0.2
599	1.00	0.989540	0.559890	0.8

600 rows × 4 columns

In [30]:
```
final_prediction
```

Out[30]:
```
0      0.885262
1      0.120823
2      0.818783
3      0.750353
4      0.564947
         ...
595    0.126340
596    0.115345
597    0.874197
598    0.163217
599    0.837357
Length: 600, dtype: float64
```

In [31]:
```
#### Calculate the ROc Curve


fpr, tpr, thresholds = roc_curve(y_test, final_prediction)
thresholds
```

Out[31]:
```
array([1.91188114, 0.91188114, 0.90406694, 0.90327475, 0.79949934,
       0.79912833, 0.79131489, 0.7905558 , 0.78597738, 0.78571156,
       0.76795305, 0.76537124, 0.74836354, 0.74637362, 0.70721721,
       0.69893711, 0.6692442 , 0.66493537, 0.5965152 , 0.59614346,
       0.58396627, 0.56800386, 0.56212652, 0.56175354, 0.55149047,
       0.54877948, 0.54355932, 0.53719563, 0.52615858, 0.49616892,
       0.4396034 , 0.38170009, 0.37629719, 0.35840767, 0.35586612,
       0.2321341 , 0.23140421, 0.21972207, 0.21896893, 0.21457968,
       0.21098417, 0.12303857, 0.1228351 , 0.10498954])
```

In [32]:
```
from sklearn.metrics import accuracy_score
accuracy_ls = []
for thres in thresholds:
    y_pred = np.where(final_prediction>thres,1,0)
    accuracy_ls.append(accuracy_score(y_test, y_pred, normalize=True))

accuracy_ls = pd.concat([pd.Series(thresholds), pd.Series(accuracy_ls)],
                        axis=1)
accuracy_ls.columns = ['thresholds', 'accuracy']
accuracy_ls.sort_values(by='accuracy', ascending=False, inplace=True)
accuracy_ls.head()
```

Out[32]:

	thresholds	accuracy
30	0.439603	0.961667
29	0.496169	0.958333
27	0.537196	0.958333
25	0.548779	0.958333
24	0.551490	0.958333

In [33]:
```
accuracy_ls
```

Out[33]:

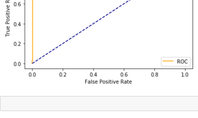
	thresholds	accuracy
30	0.439603	0.961667
29	0.496169	0.958333
27	0.537196	0.958333
25	0.548779	0.958333
24	0.551490	0.958333
28	0.526159	0.956667
26	0.543559	0.956667
23	0.561754	0.955000
22	0.562127	0.953333
19	0.596143	0.951667
21	0.568004	0.951667
20	0.583966	0.951667
18	0.596515	0.950000
31	0.381700	0.946667
32	0.376297	0.945000
33	0.358408	0.945000
34	0.355866	0.943333
17	0.664935	0.935000
16	0.669244	0.933333
15	0.698937	0.923333
14	0.707217	0.921667
13	0.746374	0.880000
12	0.748364	0.878333
35	0.232134	0.873333
36	0.231404	0.871667
37	0.219722	0.860000
11	0.765371	0.860000
38	0.218969	0.858333
10	0.767953	0.858333
39	0.214580	0.850000
40	0.210984	0.848333
9	0.785712	0.836667
8	0.785977	0.835000
7	0.790556	0.828333
6	0.791315	0.826667
5	0.799128	0.820000
4	0.799499	0.818333
41	0.122039	0.680000
42	0.122835	0.678333
3	0.903275	0.515000
2	0.904067	0.513333
43	0.104990	0.501667
1	0.911881	0.500000
0	1.911881	0.500000

In [34]:
```
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```

In [35]:
```
plot_roc_curve
```

Out[35]:
```
<function __main__.plot_roc_curve(fpr, tpr)>
```

In [36]:
```
plot_roc_curve(fpr,tpr)
```



In [ ]:
```

```