Select the Right Threshold values using ROC Curve

In [1]:
```
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 [2]:
```
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 [3]:
```
# weight ka matlb hai ki 2 classes equal no me honge 1000-1000 (balanced)
```

In [4]:
```
X.shape
```

Out[4]: (2000, 20)

In [5]:
```
y
```

Out[5]: array([0, 0, 0, ..., 1, 1, 0])

In [6]:
```
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 [7]:
```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

Logistic Regression

In [8]:
```
from sklearn.linear_model import LogisticRegression
log_classifier=LogisticRegression()
log_classifier.fit(X_train, y_train)
```

Out[8]: LogisticRegression()

In [9]:
```
ytrain_pred = log_classifier.predict_proba(X_train) # yaha th =0.5 (default) (th = - Threshold)
'''predict_proba , ye probabilty btayega jb classification 2 classes ki hai toh ek manlo 0 or dusra
toh kisi feature k basis pe class predict krni hai toh 0 or 1 ane ki probability nikal ke dega
niche dekho 2 column hai , ye jb th default 0.5 ho tb '''
```

Out[9]: 'predict_proba , ye probabilty btayega jb classification 2 classes ki hai toh ek manlo 0 or dusra 1\n toh kisi feature k basis pe class predict krni hai toh 0 or 1 ane ki probability nikal ke dega \nniche d ekho 2 column hai , ye jb th default 0.5 ho tb '

In [10]:
```
ytrain_pred.shape # 2 column isliye h , ek btayega probabilty of being 0 or ek 1
```

Out[10]: (1400, 2)

In [11]:
```
ytrain_pred
```

Out[11]:
```
array([[9.99688479e-01, 3.11520635e-04],
       [9.99966000e-01, 3.39995695e-05],
       [2.09976771e-02, 9.79002323e-01],
       ...,
       [8.58463348e-01, 1.41536652e-01],
       [9.99422335e-01, 5.77665099e-04],
       [6.07714035e-01, 3.92285965e-01]])
```

In [12]:
```
ytrain_pred[:,1]
```

Out[12]:
```
array([3.11520635e-04, 3.39995695e-05, 9.79002323e-01, ...,
       1.41536652e-01, 5.77665099e-04, 3.92285965e-01])
```

In [13]:
```
print('Logistic train roc-auc: {}'.format(roc_auc_score(y_train, ytrain_pred[:,1])))
# ye probabilty hai "1"  ki [:,1]
# roc  auc  score shows area under curve in percentage
```

Logistic train roc-auc: 0.9863568922694498

In [14]:
```
ytest_pred = log_classifier.predict_proba(X_test)
print('Logistic test roc-auc: {}'.format(roc_auc_score(y_test, ytest_pred[:,1])))
```

Logistic test roc-auc: 0.9885777777777777

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

In [15]:
```
pred=[]
pred.append(pd.Series(log_classifier.predict_proba(X_test)[:,1]))
```

In [16]:
```
pred
```

Out[16]:
```
[0      0.991861
 1      0.000008
 2      0.966929
 3      0.761539
 4      0.779443
         ...
 595    0.024239
 596    0.000003
 597    0.984385
 598    0.001147
 599    0.989540
 Length: 600, dtype: float64]
```

In [17]:
```
'''nihce ka ye unnessary hai , vo bot sare model hoto nihce wala use krege jisme , mean
nikal k karenge '''
```

Out[17]: 'nihce ka ye unnessary hai , vo bot sare model hoto nihce wala use krege jisme , mean \nnikal k k arenge '

In [18]:
```
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: 0.9885777777777777

In [29]:
```
pd.concat(pred,axis=1) #these are prediction wrt "1"
```

Out[29]:

	0
0	0.991861
1	0.000008
2	0.966929
3	0.761539
4	0.779443
...	...
595	0.024239
596	0.000003
597	0.984385
598	0.001147
599	0.989540

600 rows × 1 columns

In [20]:
```
final_prediction
```

Out[20]:
```
0      0.991861
1      0.000008
2      0.966929
3      0.761539
4      0.779443
         ...
595    0.024239
596    0.000003
597    0.984385
598    0.001147
599    0.989540
Length: 600, dtype: float64
```

In [21]:
```
#### Calculate the ROc Curve


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

Out[21]:
```
array([1.99970150e+00, 9.99701500e-01, 9.96158877e-01, 9.96129645e-01,
       9.47070326e-01, 9.46204924e-01, 8.65466258e-01, 8.63536252e-01,
       8.53176377e-01, 8.50056757e-01, 8.41421435e-01, 8.39367909e-01,
       8.15506733e-01, 8.14031083e-01, 7.10421057e-01, 6.95370907e-01,
       6.71015565e-01, 6.37604614e-01, 6.28000190e-01, 6.25419393e-01,
       5.85991638e-01, 5.72811301e-01, 5.44222421e-01, 5.09091565e-01,
       5.05747727e-01, 4.25206094e-01, 4.00497635e-01, 3.57672321e-01,
       3.57418343e-01, 3.08833885e-01, 3.04354181e-01, 2.98609914e-01,
       2.96733938e-01, 2.62534344e-01, 2.58894947e-01, 2.46055520e-01,
       2.13787155e-01, 8.32534990e-02, 8.12384385e-02, 5.22202002e-06,
       4.99437632e-06, 2.17237065e-07])
```

In [22]:
```
fpr
```

Out[22]:
```
array([0.        , 0.        , 0.        , 0.00333333, 0.00333333,
       0.00666667, 0.00666667, 0.01      , 0.01      , 0.01333333,
       0.01333333, 0.01666667, 0.01666667, 0.02      , 0.02      ,
       0.02333333, 0.02333333, 0.03      , 0.03      , 0.03333333,
       0.03333333, 0.03666667, 0.03666667, 0.04      , 0.04      ,
       0.05666667, 0.05666667, 0.06333333, 0.06333333, 0.07666667,
       0.07666667, 0.08      , 0.08      , 0.09      , 0.09      ,
       0.1       , 0.1       , 0.17      , 0.17      , 0.95666667,
       0.95666667, 1.        ])
```

In [23]:
```
tpr
```

Out[23]:
```
array([0.        , 0.00333333, 0.11      , 0.11      , 0.65666667,
       0.65666667, 0.81      , 0.81      , 0.83      , 0.83      ,
       0.84333333, 0.84333333, 0.87333333, 0.87333333, 0.93666667,
       0.93666667, 0.95      , 0.95      , 0.95333333, 0.95333333,
       0.96333333, 0.96333333, 0.97      , 0.97      , 0.97333333,
       0.97333333, 0.97666667, 0.97666667, 0.98      , 0.98      ,
       0.98333333, 0.98333333, 0.98666667, 0.98666667, 0.99      ,
       0.99      , 0.99333333, 0.99333333, 0.99666667, 0.99666667,
       1.        , 1.        ])
```

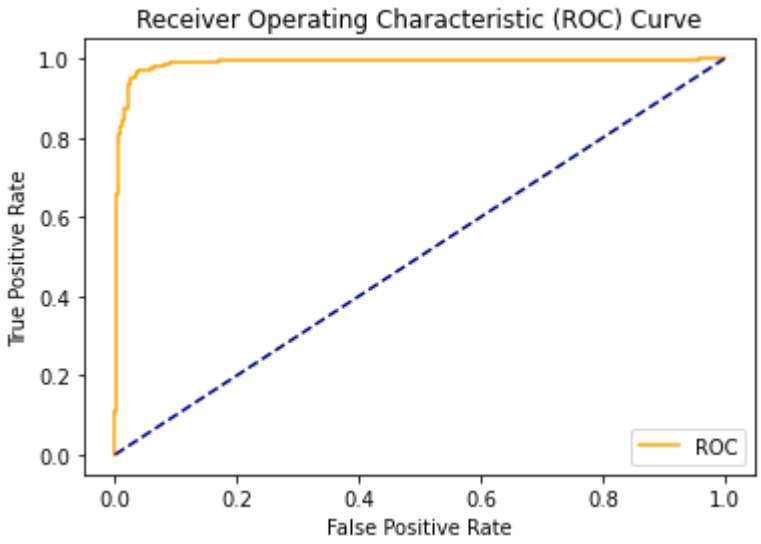
In [24]:
```
from sklearn.metrics import accuracy_score
accuracy_ls = []
for thres in thresholds:
    y_pred = np.where(final_prediction>thres,1,0)
    accuracy_ls.append(accuracy_score(y_test, y_pred, normalize=True))

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

Out[24]:

	thresholds	accuracy	fpr	tpr
23	0.509092	0.966667	0.040000	0.970000
21	0.572811	0.965000	0.036667	0.963333
24	0.505748	0.965000	0.040000	0.973333
22	0.544222	0.965000	0.036667	0.970000
20	0.585992	0.963333	0.033333	0.963333

In [25]:
```
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 [26]:
```
plot_roc_curve(fpr,tpr)
```



In [ ]:
```

```