In [12]:

```
weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny','Rainy','Sunny','O
temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']

play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']
```

In [13]:

```
# Encoding Features
# Import LabelEncoder
from sklearn import preprocessing
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
weather_encoded=le.fit_transform(weather)
print (weather_encoded)
```

```
[2 2 0 1 1 1 0 2 2 1 2 0 0 1]
```

In [14]:

```
# Converting string labels into numbers
temp_encoded=le.fit_transform(temp)
label=le.fit_transform(play)
print ("Temp:",temp_encoded)
print ("Play:",label)
```

```
Temp: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]
Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

In [15]:

```
features = list(zip(weather_encoded, temp_encoded))
```

In [16]:

```
features
```

Out[16]:

```
[(2, 1),
 (2, 1),
 (0, 1),
 (1, 2),
 (1, 0),
 (1, 0),
 (0, 0),
 (2, 2),
 (2, 0),
 (1, 2),
 (2, 2),
 (0, 2),
 (0, 1),
 (1, 2)]
```

Generating Model

In [17]:

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
model = GaussianNB()
```

In [18]:

```
# Train the model using the training sets
model.fit(features, label)
```

Out[18]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [19]:

```
predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
print ("Predicted Value:", predicted)
```

```
Predicted Value: [1]
```

With Multiple Lables

In [20]:

```
import numpy as np
import pandas as pd
import seaborn as sns
sns.set(color_codes=True)
import matplotlib.pyplot as plt
%matplotlib inline
```

In [21]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

In [22]:

```
pima_df = pd.read_csv(r"C:\Users\ABHISHEK\Desktop\diabetes.csv")
pima_df.head()
```

Out[22]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [23]:

```
pima_df.isnull().sum()
```

Out[23]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [24]:

```
X = pima_df.drop("Outcome", axis = 1)
Y = pima_df[ ["Outcome"] ]
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size= 0.2, random_state = 1)
```

In [25]:

```
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
Y_pred
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConve
rsionWarning: 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().
  y = column_or_1d(y, warn=True)
```

Out[25]:

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

In [26]:

```
from sklearn import metrics

# make predictions
predicted = model.predict(X_test)
from sklearn.metrics import accuracy_score, confusion_matrix
metrics.confusion_matrix(predicted, Y_test)
```

Out[26]:

```
array([[85, 21],
       [14, 34]], dtype=int64)
```

In [27]:

```
model_score = model.score(X_test, Y_test)
model_score
```

Out[27]:

```
0.7727272727272727
```

In [28]:

```
y_predictProb = model.predict_proba(X_test)
from sklearn.metrics import auc, roc_curve
fpr, tpr, thresholds = roc_curve(Y_test, y_predictProb[::,1])
roc_auc = auc(fpr, tpr)
roc_auc
```

Out[28]:

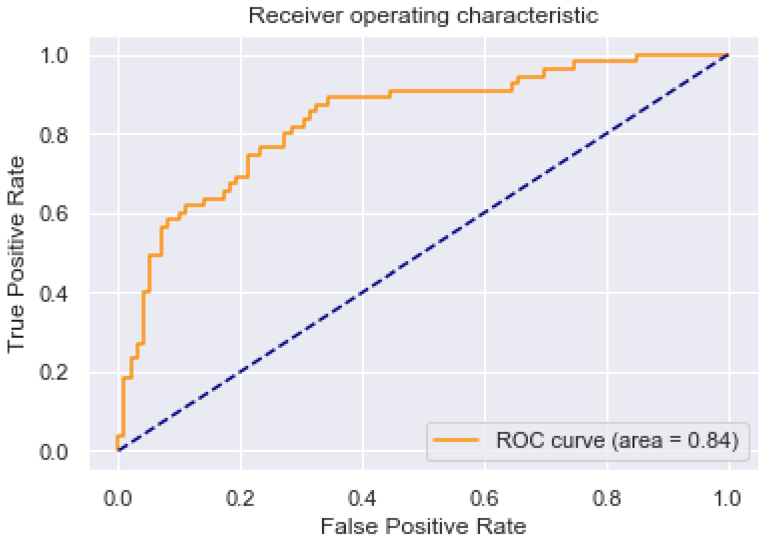
```
0.8359963269054177
```

In [29]:

```
plt.plot(fpr, tpr, color='darkorange', label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
```

Out[29]:

```
<matplotlib.legend.Legend at 0x153242518d0>
```


Receiver operating characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.84)