M.Nilofer Sultana,

AP17110010058,

CSE-01.

Implementing Navïe Bayes from scratch and calcualating the accuracy, precision.

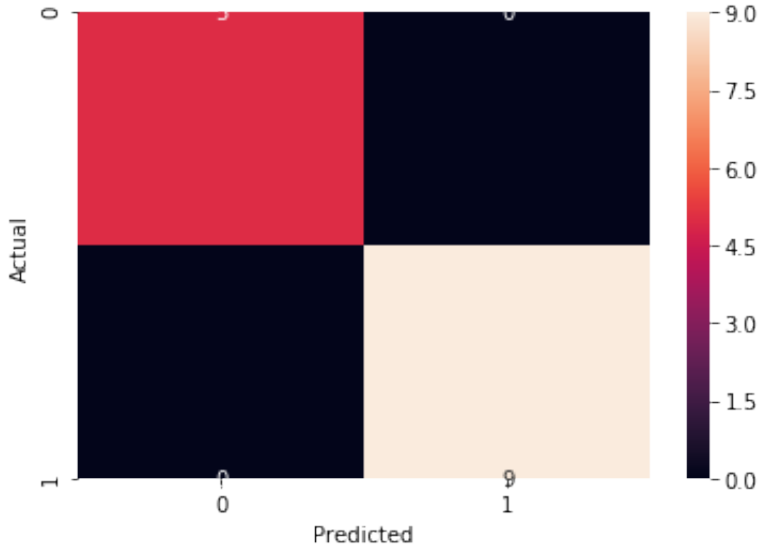
In [4]:
```
import numpy as np
import pandas as pd
from sklearn import datasets
data = pd.read_csv("tennis.csv")
data.columns
data.head(14)
X_train = pd.get_dummies(data[['outlook', 'temp', 'humidity', 'wind
y_train = pd.DataFrame(data['play'])

outlook_count = data.groupby(['outlook', 'play']).size()
outlook_total = data.groupby(['outlook']).size()
temp_count = data.groupby(['temp', 'play']).size()
temp_total = data.groupby(['temp']).size()
humidity_count = data.groupby(['humidity', 'play']).size()
humidity_total = data.groupby(['outlook']).size()
windy_count = data.groupby(['windy', 'play']).size()
windy_total = data.groupby(['windy']).size()
play_count=data.groupby(['play']).size()

print(outlook_count)
print(windy_total)
print(outlook_total)
print(temp_count)
print(temp_total)
print(humidity_count)
print(humidity_total)
print(windy_count)
print(windy_total)

p_total=14
p_over_yes = outlook_count['overcast','yes']
p_rainy_yes = outlook_count['rainy','yes']
p_sunny_yes = outlook_count['sunny', 'yes']
p_hot_yes=temp_count['hot','yes']
p_humid_yes=humidity_count['normal','yes']
p_windy_yes=windy_count['weak','yes']
p_yes=play_count['yes']
p_rainy_no = outlook_count['rainy','no']
p_sunny_no = outlook_count['sunny', 'no']
```

```
p_sunny_no = outlook_count['sunny','no']
p_hot_no=temp_count['hot','no']
p_humid_no=humidity_count['normal','no']
p_windy_no=windy_count['weak','no']
p_no=play_count['no']

P_play=(p_sunny_yes/p_yes)*(p_hot_yes/p_yes)*(p_windy_yes/p_yes)*(p
print("the probability to play: ",P_play)

P_play_no=(p_sunny_no/p_no)*(p_hot_no/p_no)*(p_windy_no/p_no)*(p_hu
print("the probability not to play: ",P_play_no)

p_total_play=(P_play+P_play_no)
P_Yes=(P_play/p_total_play)
P_No=(P_play_no/p_total_play)

print("Total probability to play: ",P_Yes)
print("Total probability not to play: ",P_No)

TP = (p_yes)
TN = (p_no)
FP = 0
FN = 0

Accuracy = (TP + TN)/(TP + TN + FP + FN)
print("Accuracy:",Accuracy)
Error = (FN + FP)/(TP + TN + FP + FN)
print("Error:",Error)
Precision = (TP)/(TP + FP )
print("Precision:", Precision)
Recall = (TP)/(TP + FN )
print("Recall:", Recall)

import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt

data = {'y_Predicted': [0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0],
        'y_Actual':    [0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0]
        }

df = pd.DataFrame(data, columns=['y_Actual','y_Predicted'])
confusion_matrix = pd.crosstab(df['y_Actual'], df['y_Predicted'], r

sn.heatmap(confusion_matrix, annot=True)
```

```
outlook   play
overcast  yes     4
rainy     no      2
          yes     3
sunny     no      3
          yes     2
dtype: int64
windy
```

```
windy
strong    6
weak      8
dtype: int64
outlook
overcast    4
rainy       5
sunny       5
dtype: int64
temp  play
cool  no      1
      yes     3
hot   no      2
      yes     2
mild  no      2
      yes     4
dtype: int64
temp
cool    4
hot     4
mild    6
dtype: int64
humidity  play
high      no      4
          yes     3
normal    no      1
          yes     6
dtype: int64
outlook
overcast    4
rainy       5

sunny       5
dtype: int64
windy   play
strong  no      3
        yes     3
weak    no      2
        yes     6
dtype: int64
windy
strong    6
weak      8
dtype: int64
the probability to play:  0.014109347442680773
the probability not to play:  0.006857142857142858
Total probability to play:  0.6729475100942126
Total probability not to play:  0.3270524899057874
Accuracy: 1.0
Error: 0.0
Precision: 1.0
Recall: 1.0
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x10e270d10>



Implementing Navïe Bayes on iris dataset and calculating accuracy and precision.

In [15]:
```
from sklearn.datasets import load_iris
iris = load_iris()


X = iris.data
y = iris.target


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

cnf_mat=[[0]*3]*3
print(cnf_mat)

y_pred = gnb.predict(X_test)

from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accurac
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
results=confusion_matrix(y_test,y_pred)
print(results)
print(classification_report(y_test,y_pred))
```

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
Gaussian Naive Bayes model accuracy(in %): 95.0
[[19  0  0]
 [ 0 19  2]
 [ 0  1 19]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.95      0.90      0.93        21
           2       0.90      0.95      0.93        20

    accuracy                           0.95        60
   macro avg       0.95      0.95      0.95        60
weighted avg       0.95      0.95      0.95        60
```