Exercise (given with NB_classifier_weather file)

(1) Will you play if the temperature is 'Hot' and weather is 'overcast'?

(2) Will you play if the temperature is 'Mild' and weather is 'Sunny'?

```
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB, MultinomialNB
weather = ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy','Rainy', 'Overcast',
           'Sunny', 'Sunny', 'Rainy', 'Sunny', 'Overcast', 'Overcast', 'Rainy']

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']

le = preprocessing.LabelEncoder()
weather_encoded=le.fit_transform(weather)

temp_encoded=le.fit_transform(temp)
label=le.fit_transform(play)

features=tuple(zip(weather_encoded,temp_encoded))
print("Features:",features)

model=MultinomialNB()
model.fit(features,label)

predicted= model.predict([[0,1]]) # 0:Overcast, 1:Hot
print("Predicted Value:", predicted)

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

```
Features: ((2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2), (2, 0), (1, 2), (2, 2)
Predicted Value: [1]
Predicted Value: [1]
```

Task 1: Try the algo on Dataset3 - LabelEncoding of features:and Train test Division 95%-5%

```
#task 1
#Import scikit-learn dataset library
import pandas as pd
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
# Import label encoder
```

```
from sklearn import preprocessing

df = pd.read_csv('/content/Dataset3.csv')

#creating labelEncoder
le = preprocessing.LabelEncoder()

# Converting string labels into numbers.
df['Outlook']=le.fit_transform(df['Outlook'])
print("Outlook:" ,df['Outlook'])

df['Temp']=le.fit_transform(df['Temp'])
print("Temp:" ,df['Temp'])

df['Wind']=le.fit_transform(df['Wind'])
print("Wind:" ,df['Wind'])
```

```
    Outlook: 0     1
    1     1
    2     0
    3     1
    4     2
```

```
#Combinig weather and temp into single listof tuples
features=tuple(zip(df['Outlook'],df['Temp'],df['Wind'],df['Humidity']))
print("Features:",features)
```

```
Features: ((1, 1, 0, 1), (1, 1, 1, 2), (0, 1, 0, 1), (1, 2, 0, 1), (2, 0, 0, 1), (0, 0, 1, 0), (0
```

```
    13    2
```

```
#(1) What will be the value of Play, if Outlook is ’Rainy’, Temperature is ’Mild’, Humidity =’Normal’,

model=MultinomialNB()
model.fit(features,label)
predicted= model.predict([[1,2,0,1]])
print("Predicted Value:", predicted)
#(2) What will be the value of Play, if Outlook is ’Sunny’, Temeprature is ’Cool’, Humidity =’High’, a
predicted= model.predict([[2,0,2,1]])
print("Predicted Value:", predicted)
```

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

```
    13    2
```

```
#import the necessary module
from sklearn.model_selection import train_test_split

#split data set into train and test sets
data=df[df.columns[0:4]]
data_train, data_test, target_train, target_test = train_test_split(data,
                        df['Class'], test_size = 0.05, random_state = 142)

print(data_train)
print('--------------')
print(target_train)
print('--------------')
print(data_test)
print('--------------')
print(target_test)
```

```
     Outlook  Temp  Wind  Humidity
8          0     0     0         0
9          2     2     0         2
7          1     2     0         1
3          1     2     0         1
4          2     0     0         1
2          0     1     0         1
12         0     1     0         1
1          1     1     1         2
0          1     1     0         1
10         1     0     1         2
11         0     2     1         0
13         2     2     1         1
5          0     0     1         0
--------------
8     1
9     1
7     0
3     1
4     1
2     1
12    1
1     0
```

```
import numpy as np
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(data_train, target_train)

#Predict the response for test dataset
target_pred = gnb.predict(data_test)
print(target_pred)
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(target_test, target_pred))
```

```
[0]
Accuracy: 0.0
```

```
#Import confusion_matrix from scikit-learn metrics module for confusion_matrix

from sklearn.metrics import confusion_matrix
confusion_matrix(target_test, target_pred)
```

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

```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

precision = precision_score(target_test, target_pred, average=None)
recall = recall_score(target_test, target_pred, average=None)
```

```
print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
```

```
precision: [0. 0.]
recall: [0. 0.]
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWa
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWa
  _warn_prf(average, modifier, msg_start, len(result))
```