Machine Learning Lab2

July 13, 2019

1 Aim: Implement Naive Bayes classifier

1.1 Objective:

```
- implement Naive bayes classifier using scikit learn library
- Test the classifier in the following dataset,
        Dataset1: Weather example discussed in the class
        Dataset2: Wine dataset, available in scikit learn
```

1.2 Procedure (Weather Example)

1.2.1 Step 1: Import necessary libraries.

We will use preprocessing libraries of sklearn and 'GaussianNB' as the classifier. (More about gaussianNB later)

[2]:
```
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
```

1.2.2 Step 2: Prepare dataset.

```
Create feature set for weather and temperature, and classlabel play.
```

[3]:
```
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']
```

1.2.3 Step 3: Digitize the data set using encoding

[4]:
```
#creating labelEncoder
le = preprocessing.LabelEncoder()

# Converting string labels into numbers.
weather_encoded=le.fit_transform(weather)
```

```
print(weather_encoded)

temp_encoded=le.fit_transform(temp)

label=le.fit_transform(play)

print("Temp:",temp_encoded)
print("Play:",label)
```

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

1.2.4 Step 4: Merge different features to prepare dataset

[5]:
```
#Combinig weather and temp into single listof tuples
features=tuple(zip(weather_encoded,temp_encoded))
print(features)
```

```
((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))
```

1.2.5 Step 5: Train 'Gaussian Naive Bayes Classifier'

[6]:
```
#Create a Gaussian Classifier
model = GaussianNB()

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

[6]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

1.2.6 Step 6: Predict Output for new data

[7]:
```
#Predict Output
predicted= model.predict([[2,2]]) # 0:Overcast, 2:Mild
print("Predicted Value:", predicted)
```

```
Predicted Value: [0]
```

1.3 Exercise

1.3.1 Manually calculate output for the following cases and compare it with system's output.

(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'?

1.3.2 Write program for Naive Bayes classifier for the data (classroom assignment) given below,

Features:
Outlook = [Rainy, Rainy, Overcast, Sunny, Sunny, Sunny, Overcast, Rainy, Rainy, Sunny, Rainy, Overcast, Overcast, Sunny]
Temperature = [Hot, Hot, Hot, Mild, Cool, Cool, Cool, Mild, Cool, Mild, Mild, Mild, Hot, Mild]
Humidity = [High, High, High, High, Normal, Normal, Normal, High, Normal, Normal, Normal, High, Normal, High]
Wind = [False, True, False, False, False, True, True, False, False, False, True, True, False, True]
Class Label:
Play = [No, No, Yes, Yes, Yes, No, Yes, No, Yes, Yes, Yes, Yes, Yes, No]
Compare System's output with Manually Calculated output for the following cases,

(1) What will be the value of Play, if Outlook is 'Rainy', Temperature is 'Mild', Humidity = 'Normal', and Wind = 'False'?

(2) What will be the value of Play, if Outlook is 'Sunny', Temeprature is 'Cool', Humidity = 'High', and Wind = 'True'?

1.4 Procedure (Wine Dataset)

1.4.1 Step 1: Load inbuilt dataset

[8]:
```
#Import scikit-learn dataset library
from sklearn import datasets

#Load dataset
wine = datasets.load_wine()
```

1.4.2 Step 2 : Check dataset

[11]:
```
# print the names of the 13 features
print("Features: ", wine.feature_names)

# print the label type of wine(class_0, class_1, class_2)
print("Labels: ", wine.target_names)

# print data(feature)shape
wine.data.shape
```

```
Features:  ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium',
'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']
Labels:  ['class_0' 'class_1' 'class_2']
```

[11]:
```
(178, 13)
```

1.4.3 Step 3 : Splitting dataset for training and testing

[19]:
```
#import the necessary module
from sklearn.model_selection import train_test_split
#split data set into train and test sets
data_train, data_test, target_train, target_test = train_test_split(wine.data,␣
 ↪wine.target, test_size = 0.30, random_state = 10)
```

1.4.4 Step 4 : Model Traning and Testing

[26]:
```
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)
```

1.4.5 Step 5 : Calculating accuracy

[27]:
```
#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))
```

```
Accuracy: 0.8888888888888888
```

1.4.6 Step 6 : Observing Confusion Matrix

[34]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(target_test, target_pred)
```

[34]:
```
array([[14,  1,  0],
       [ 2, 22,  3],
       [ 0,  0, 12]])
```

1.5 Exercise

1.5.1 1. Build a naive bayes classifier to categorize Iris dataset into

(1) Three categories : Iris (Iris setosa, Iris virginica and Iris versicolor)
(2) Two categories : Iris (Iris virginica and Iris versicolor)
generate confusin matrix and calculate precision and recall in both cases.
(Iris Dataset Ref: https://en.wikipedia.org/wiki/Iris_flower_data_set)