Aim: Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm

Short notes: Naive Bayes

Bayes' Theorem provides a way that we can calculate the probability of a piece of data belonging to a given class, given our prior knowledge. Bayes' Theorem is stated as:

P(class|data) = (P(data|class) * P(class)) / P(data)

Where P(class|data) is the probability of class given the provided data.

We are using Iris Dataset. The Iris Flower Dataset involves predicting the flower species given measurements of iris flowers.

It is a multiclass classification problem. The number of observations for each class is balanced. There are 150 observations with 4 input variables and 1 output variable. The variable names are as follows:

Sepal length in cm.

Sepal width in cm.

Petal length in cm.

Petal width in cm., and

Class.

Algorithm:

Step 1: Separate By Class.

Step 2: Summarize Dataset.

Step 3: Summarize Data By Class.

Step 4: Gaussian Probability Density Function.

Step 5: Class Probabilities.


```
#Import Modules
#Load iris dataset & do train_test_split
from sklearn import neighbors,datasets,preprocessing
from sklearn .model_selection import train_test_split
iris=datasets.load_iris()
x=iris.data[:,:]
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,stratify=y,random_state=42)
```

```
#Feature Scaling
```

```
2 scaler=preprocessing.StandardScaler().fit(x_train)
3 x_train=scaler.transform(x_train)
4 x_test=scaler.transform(x_test)
5 x_train
```

```
array([[ 1.79213839, -0.60238047,  1.31532306,  0.92095427],
       [ 2.14531053, -0.60238047,  1.65320421,  1.05135487],
       [-0.4446185 , -1.50797259, -0.03620155, -0.25265117],
       [ 0.26172578, -0.60238047,  0.13273902,  0.13855064],
       [-0.4446185 , -1.28157456,  0.13273902,  0.13855064],
       [ 0.49717388,  0.52960968,  1.25900953,  1.70335789],
       [-1.50413492,  0.75600771, -1.33141264, -1.1654554 ],
       [ 0.49717388, -0.8287785 ,  0.63956075,  0.79055366],
       [-1.26868682,  0.07681362, -1.21878559, -1.295856  ],
       [ 0.37944983, -0.60238047,  0.58324723,  0.79055366],
       [-0.91551468,  1.66159983, -1.04984501, -1.03505479],
       [ 0.61489792, -0.8287785 ,  0.86481486,  0.92095427],
       [-1.03323873, -2.41356471, -0.1488286 , -0.25265117],
       [-0.4446185 ,  2.56719194, -1.33141264, -1.295856  ],
       [-0.79779064,  2.34079391, -1.27509911, -1.42625661],
       [-0.09144636, -0.8287785 ,  0.0764255 ,  0.00815004],
       [ 1.55669029, -0.14958441,  1.14638248,  0.52975245],
       [-0.91551468,  0.98240574, -1.33141264, -1.1654554 ],
       [-0.2091704 ,  3.019988  , -1.27509911, -1.03505479],
       [-0.79779064, -0.8287785 ,  0.0764255 ,  0.26895125],
       [ 1.3212422 ,  0.30321165,  0.5269337 ,  0.26895125],
       [ 1.20351815,  0.07681362,  0.63956075,  0.39935185],
       [-0.68006659,  1.4352018 , -1.27509911, -1.295856  ],
       [-0.32689445, -0.37598244, -0.09251508,  0.13855064],
       [ 0.96807006, -0.14958441,  0.80850133,  1.44255668],
       [-1.15096278, -0.14958441, -1.33141264, -1.295856  ],
       [ 1.08579411,  0.30321165,  1.20269601,  1.44255668],
       [ 0.49717388, -1.73437062,  0.35799313,  0.13855064],
       [ 1.20351815,  0.30321165,  1.09006896,  1.44255668],
       [-1.03323873,  1.20880377, -1.33141264, -1.295856  ],
       [-1.15096278,  0.07681362, -1.27509911, -1.295856  ],
       [ 0.96807006,  0.07681362,  0.5269337 ,  0.39935185],
       [-1.15096278,  0.07681362, -1.27509911, -1.42625661],
       [ 0.49717388, -0.60238047,  0.7521878 ,  0.39935185],
       [-1.15096278,  1.20880377, -1.33141264, -1.42625661],
       [-1.26868682,  0.75600771, -1.04984501, -1.295856  ],
       [-0.32689445, -0.14958441,  0.18905255,  0.13855064],
       [-0.91551468,  0.75600771, -1.27509911, -1.295856  ],
       [ 0.14400174, -0.37598244,  0.41430665,  0.39935185],
       [-1.85730706, -0.14958441, -1.50035321, -1.42625661],
       [-1.26868682, -0.14958441, -1.33141264, -1.42625661],
       [-0.2091704 , -0.14958441,  0.24536608,  0.00815004],
       [-0.09144636, -0.60238047,  0.7521878 ,  1.57295728],
       [-0.09144636, -0.8287785 ,  0.7521878 ,  0.92095427],
       [-1.03323873, -0.14958441, -1.21878559, -1.295856  ],
       [ 0.73262197,  0.30321165,  0.7521878 ,  1.05135487],
       [ 0.61489792, -0.37598244,  0.3016796 ,  0.13855064],
       [-0.91551468,  1.66159983, -1.27509911, -1.1654554 ],
       [ 1.08579411, -0.14958441,  0.97744191,  1.18175547],
       [-1.38641087,  0.30321165, -1.21878559, -1.295856  ],
       [ 0.61489792, -0.60238047,  1.03375543,  1.18175547],
       [ 2.14531053,  1.66159983,  1.65320421,  1.31215608],
       [-0.56234254,  1.88799786, -1.38772616, -1.03505479],
       [ 0.73262197, -0.60238047,  0.47062018,  0.39935185],
```

```
[ 0.49717388, -1.28157456,  0.69587428,  0.92095427],
[ 2.38075862,  1.66159983,  1.48426364,  1.05135487],
[ 0.96807006,  0.07681362,  0.35799313,  0.26895125],
```

In this step, we introduce the class GaussianNB that is used from the sklearn.naive_bayes library. Here, we have used a Gaussian model, there are several other models such as Bernoulli, Categorical and Multinomial. Here, we assign the GaussianNB class to the variable classifier and fit the X_train and y_train values to it for training purpose.

```
#Implement Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
classifer=GaussianNB()
classifer.fit(x_train,y_train)


```

```
GaussianNB()
```

```
#Predict the values for test data
y_pred=classifer.predict(x_test)

# Display accuracy score & display confusion matrix & classification report
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
0.9210526315789473
[[12  0  0]
 [ 0 12  1]
 [ 0  2 11]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.86      0.92      0.89        13
           2       0.92      0.85      0.88        13

    accuracy                           0.92        38
   macro avg       0.92      0.92      0.92        38
weighted avg       0.92      0.92      0.92        38
```

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
classifer=BernoulliNB()
classifer.fit(x_train,y_train)
y_pred=classifer.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))

```

```
0.7894736842105263
```

```
[[12  0  0]
 [ 2  6  5]
 [ 0  1 12]]
              precision    recall  f1-score   support

           0       0.86      1.00      0.92        12
           1       0.86      0.46      0.60        13
           2       0.71      0.92      0.80        13

    accuracy                           0.79        38
   macro avg       0.81      0.79      0.77        38
weighted avg       0.81      0.79      0.77        38
```

From the above confusion matrix, we infer that, out of 45 test set data, 44 were correctly classified and only 1 was incorrectly classified. This gives us a high accuracy of 97.7%.

✓ 0s completed at 12:35 PM