EXPERIMENT – 06

Write a program to implement Naïve Bayes algorithm in python and to display the results using confusion matrix and accuracy.

CODE :

```
# Naive Bayes without sklearn

# Training data

X = [[1], [2], [3], [6], [7], [8]]  # Feature

y = [0, 0, 0, 1, 1, 1]              # Labels

# Separate by class

class0 = [X[i][0] for i in range(len(X)) if y[i] == 0]

class1 = [X[i][0] for i in range(len(X)) if y[i] == 1]


# Mean and variance for each class

def mean(lst):

    return sum(lst)/len(lst)

def var(lst):

    m = mean(lst)

    return sum((x-m)**2 for x in lst)/len(lst)

mean0, var0 = mean(class0), var(class0)

mean1, var1 = mean(class1), var(class1)

# Gaussian probability

def gaussian(x, m, v):

    import math

    return (1 / math.sqrt(2*math.pi*v)) * math.exp(-(x-m)**2/(2*v))


# Predict

def predict(x):

    prob0 = gaussian(x, mean0, var0) * (len(class0)/len(X))

    prob1 = gaussian(x, mean1, var1) * (len(class1)/len(X))
```

```
    return 0 if prob0 > prob1 else 1
# Test
X_test = [4, 7]
y_test = [0, 1]
y_pred = [predict(x) for x in X_test]
# Confusion Matrix
TP = sum(1 for i,j in zip(y_test, y_pred) if i==j==1)
TN = sum(1 for i,j in zip(y_test, y_pred) if i==j==0)
FP = sum(1 for i,j in zip(y_test, y_pred) if i==0 and j==1)
FN = sum(1 for i,j in zip(y_test, y_pred) if i==1 and j==0)
print("Predicted:", y_pred)
print("Confusion Matrix:")
print([[TN, FP],
       [FN, TP]])
accuracy = (TP + TN)/len(y_test)
print("Accuracy:", accuracy)
```

OUTPUT :

```
=== RESTART: C:\Users\lalit\AppData\Local\Programs\Python\Python313\exp-06.py ==
Predicted: [0, 1]
Confusion Matrix:
[[1, 0], [0, 1]]
Accuracy: 1.0
>>>
```