

Experiment -16

```
import math

# Simple dataset: [feature1, feature2, class]
data = [
    [1, 2, 0],
    [2, 3, 0],
    [3, 3, 1],
    [6, 5, 1]
]

test = [2, 2]
actual = 0

# ----- Naive Bayes -----
def naive_bayes():
    class0 = [d for d in data if d[-1] == 0]
    class1 = [d for d in data if d[-1] == 1]
    return 0 if len(class0) > len(class1) else 1

# ----- KNN -----
def knn(k=3):
    dist = []
    for d in data:
        dist.append((math.dist(test, d[:-1]), d[-1]))
    dist.sort()
    labels = [l for _, l in dist[:k]]
    return max(set(labels), key=labels.count)

# ----- Decision Tree (Rule-based) -----
def decision_tree():
    return 0 if test[0] < 3 else 1

# Predictions
nb = naive_bayes()
kn = knn()
dt = decision_tree()

print("Naive Bayes Prediction:", nb)
print("KNN Prediction:", kn)
print("Decision Tree Prediction:", dt)

# Accuracy
```

```
print("\nAccuracy:")
print("Naive Bayes:", int(nb == actual))
print("KNN:", int(kn == actual))
print("Decision Tree:", int(dt == actual))
```

Output:

main.py

Share

Run

```
1 import math
2 data = [
3     [1, 2, 0],
4     [2, 3, 0],
5     [3, 3, 1],
6     [6, 5, 1]
7 ]
8 test = [2, 2]
9 actual = 0
10 def naive_bayes():
11     class0 = [d for d in data if d[-1] == 0]
12     class1 = [d for d in data if d[-1] == 1]
13     return 0 if len(class0) > len(class1) else 1
14 def knn(k=3):
15     dist = []
16     for d in data:
17         dist.append((math.dist(test, d[:-1]), d[-1]))
18     dist.sort()
19     labels = [1 for _, 1 in dist[:k]]
20     return max(set(labels), key=labels.count)
21 def decision_tree():
22     return 0 if test[0] < 3 else 1
23 nb = naive_bayes()
24 kn = knn()
25 dt = decision_tree()
```

Output

Clear

```
Naive Bayes Prediction: 1
KNN Prediction: 0
Decision Tree Prediction: 0

Accuracy:
Naive Bayes: 0
KNN: 1
Decision Tree: 1

=== Code Execution Successful ===
```

Activate Windows
Go to Settings to activate Windows.