NAIVE_BAYES_CLASSIFIER

January 30, 2026

[1]:
```
"""
Naive Bayes Classifier - Categorical Data
---------------------------------------
- Multiple categorical attributes
- Categorical class labels
- Pure Python
- Robust input handling
"""

from collections import defaultdict


def train_naive_bayes(data, labels):
    class_count = defaultdict(int)
    feature_count = defaultdict(lambda: defaultdict(lambda: defaultdict(int)))

    total = len(labels)

    for row, label in zip(data, labels):
        class_count[label] += 1
        for i, value in enumerate(row):
            feature_count[label][i][value] += 1

    return class_count, feature_count, total


def predict(class_count, feature_count, total, test_point):
    probabilities = {}

    for label in class_count:
        prob = class_count[label] / total  # prior

        for i, value in enumerate(test_point):
            count = feature_count[label][i].get(value, 0)
            total_feature = sum(feature_count[label][i].values())

            if total_feature == 0:
                prob *= 0
            else:
                prob *= count / total_feature

        probabilities[label] = prob

    return max(probabilities, key=probabilities.get)


def main():
    attributes = input("Enter attribute names (space separated): ").split()
    target = input("Enter output column name (single word): ")

    while True:
        try:
            n = int(input("Enter number of records: "))
            break
        except ValueError:
            print("âİŇ Please enter an integer value.")

    data = []
    labels = []

    print("\nEnter dataset values:")
    for _ in range(n):
        row = input(f"Enter values for {attributes}: ").split()
        label = input(f"Enter {target}: ")
        data.append(row)
        labels.append(label)

    test_point = input(
        f"\nEnter values for {attributes} to predict {target}: "
    ).split()

    class_count, feature_count, total = train_naive_bayes(data, labels)
    result = predict(class_count, feature_count, total, test_point)

    print(f"\nPredicted {target}: {result}")


if __name__ == "__main__":
    main()
```

```
Enter attribute names (space separated):  Weight Height
Enter output column name (single word):  Species
Enter number of records:  8
Enter dataset values:

Enter values for ['Weight', 'Height']:  4 35
Enter Species:  Cat
Enter values for ['Weight', 'Height']:  6 40
Enter Species:  Rat
Enter values for ['Weight', 'Height']:  3 25
Enter Species:  Cat
Enter values for ['Weight', 'Height']:  7 45
Enter Species:  Rat
Enter values for ['Weight', 'Height']:  5 30
Enter Species:  Cat
Enter values for ['Weight', 'Height']:  8 50
Enter Species:  Rat
Enter values for ['Weight', 'Height']:  2 20
Enter Species:  Cat
Enter values for ['Weight', 'Height']:  5 35
Enter Species:  Rat

Enter values for ['Weight', 'Height'] to predict Species:  4 30


Predicted Species: Cat
```

```
[ ]:
```