**Code:**

import math data = [

([3.4, 5.1], 'A'),

([2.3, 4.5], 'A'),

([3.1, 4.9], 'A'),

([3.6, 5.2], 'A'),

([2.8, 4.7], 'A'),

([5.1, 1.4], 'B'),

([4.7, 1.3], 'B'),

([4.9, 1.5], 'B'),

([5.2, 1.4], 'B'),

([4.6, 1.4], 'B'),

]

train\_data = data[:8] test\_data = data[8:]

class\_counts = {}

for \_, label in train\_data:

class\_counts[label] = class\_counts.get(label, 0) + 1

total\_samples = len(train\_data)

class\_probabilities = {label: count / total\_samples for label, count in class\_counts.items()}

class\_feature\_stats = {}

for features, label in train\_data:

if label not in class\_feature\_stats:

class\_feature\_stats[label] = {'mean': [0, 0], 'variance': [0, 0]}

for i, feature in enumerate(features): class\_feature\_stats[label]['mean'][i] += feature class\_feature\_stats[label]['variance'][i] += feature \*\* 2

for label, stats in class\_feature\_stats.items(): for i in range(len(stats['mean'])): stats['mean'][i] /= class\_counts[label]

stats['variance'][i] = (stats['variance'][i] / class\_counts[label]) - (stats['mean'][i]

\*\*2)

def calculate\_probability(x, mean, variance):

exponent = math.exp(-(x - mean) \*\* 2 / (2 \* variance)) return (1 / (math.sqrt(2 \* math.pi \* variance))) \* exponent

def classify(features):

best\_class = None best\_probability = -1

for label, prior\_probability in class\_probabilities.items(): likelihood = 1

for i, feature in enumerate(features):

mean = class\_feature\_stats[label]['mean'][i] variance = class\_feature\_stats[label]['variance'][i]

likelihood \*= calculate\_probability(feature, mean, variance) posterior\_probability = prior\_probability \* likelihood

if posterior\_probability > best\_probability: best\_probability = posterior\_probability best\_class = label

return best\_class

correct\_predictions = 0 total\_predictions = len(test\_data)

for features, true\_label in test\_data: predicted\_label = classify(features)

print(f"Predicted: {predicted\_label}, True: {true\_label}")

if predicted\_label == true\_label: correct\_predictions += 1

accuracy = correct\_predictions / total\_predictions print(f"Accuracy: {accuracy \* 100:.2f}%")

**Output:**

C:/Users/admin/AppData/Local/Programs/Python/Python311/python.exe c:/Users/admin/Desktop/1020204/naive.py

Predicted: B, True: B Predicted: B, True: B Accuracy: 100.00%