

EXPERIMENT 6:

```
import math

# Dataset (features, class)
X = [
    [1, 20], [2, 22], [3, 21],
    [8, 30], [9, 32], [10, 31]
]
y = [0, 0, 0, 1, 1, 1]

# Test data
X_test = [[2,21], [9,31], [3,20], [10,33]]
y_test = [0, 1, 0, 1]

# Separate data by class
classes = set(y)
data = {c: [] for c in classes}
for xi, yi in zip(X, y):
    data[yi].append(xi)

# Mean & Variance
def mean(nums): return sum(nums)/len(nums)
def var(nums):
    m = mean(nums)
    return sum((x-m)**2 for x in nums)/len(nums)

# Gaussian probability
def gaussian(x, m, v):
    return (1/math.sqrt(2*math.pi*v)) * math.exp(-(x-m)**2/(2*v))
```

```

# Train model

stats = { }

for c in classes:
    stats[c] = []
    for col in zip(*data[c]):
        stats[c].append((mean(col), var(col)))

# Predict
def predict(x):
    probs = { }
    for c in classes:
        probs[c] = 1
        for i in range(len(x)):
            m,v = stats[c][i]
            probs[c] *= gaussian(x[i], m, v)
    return max(probs, key=probs.get)

# Predictions
y_pred = [predict(x) for x in X_test]

# Confusion Matrix
cm = [[0,0],[0,0]]
for a,p in zip(y_test,y_pred):
    cm[a][p] += 1

# Accuracy
acc = sum(cm[i][i] for i in range(2)) / sum(sum(r) for r in cm)

print("Confusion Matrix:")

```

```
for row in cm:  
    print(row)  
print("Accuracy:", acc)
```

Output

Accuracy: 1.0

=== Code Execution Successful ===