LAB 9

Naive Bayes Classifier

1. Naive Bayes (No Libraries) -

Code -

```python
# Training dataset
data = [
    ['Sunny', 'Hot', 'No'],
    ['Sunny', 'Hot', 'No'],
    ['Overcast', 'Hot', 'Yes'],
    ['Rainy', 'Mild', 'Yes'],
    ['Rainy', 'Cool', 'Yes'],
    ['Rainy', 'Cool', 'No'],
    ['Overcast', 'Cool', 'Yes'],
    ['Sunny', 'Mild', 'No'],
    ['Sunny', 'Cool', 'Yes']
]

# Count function
def count_freq(attribute_index, value, target, data):
    count = 0
    target_count = 0
    for row in data:
        if row[2] == target:
            target_count += 1
            if row[attribute_index] == value:
                count += 1
    return count, target_count

# Predict for: ['Sunny', 'Cool']
def predict(test):
    classes = ['Yes', 'No']
    probs = {}

    for cls in classes:
        prob = 1
        for i in range(len(test)):
            attr_count, total = count_freq(i, test[i], cls, data)
            prob *= (attr_count / total) if total != 0 else 0
        class_count = sum(1 for row in data if row[2] == cls)
        prob *= class_count / len(data)
        probs[cls] = prob

    return max(probs, key=probs.get)

# Run prediction
test_sample = ['Sunny', 'Cool']
print("Prediction for", test_sample, "=>", predict(test_sample))
test_sample = ['Overcast', 'Mild']
print("Prediction for", test_sample, "=>", predict(test_sample))
test_sample = ['Rainy', 'Hot']
print("Prediction for", test_sample, "=>", predict(test_sample))
test_sample = ['Sunny', 'Mild']
print("Prediction for", test_sample, "=>", predict(test_sample))
test_sample = ['Rainy', 'Cool']
print("Prediction for", test_sample, "=>", predict(test_sample))
```

Output -

```
Prediction for ['Sunny', 'Cool'] => No
Prediction for ['Overcast', 'Mild'] => Yes
Prediction for ['Rainy', 'Hot'] => No
Prediction for ['Sunny', 'Mild'] => No
Prediction for ['Rainy', 'Cool'] => Yes
```

2. Using python libraries with Dataset

Code -

```python
# Load data and train/test a Naïve Bayes Classifier
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load dataset
df = pd.read_csv("Mall_Customers.csv")

# Encode 'Gender' to numeric
df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})

# Select features and target
X = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
y = df['Gender']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5,
random_state=45)

# Train Naive Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Accuracy
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy: {acc * 100:.2f}%")
```

Output - Accuracy: 65.00%