Name: **Gargi Devendra Shintre**
Div: **D15C** Roll No. **69**

Practical No. 4

Aim:

To implement classification algorithms: Decision Tree and Naïve Bayes using Python.

Theory:

Classification in Machine Learning:

Classification is a supervised learning technique where the model learns from labeled data and predicts the category (class) of new input data. In this experiment, the goal is to classify products into their respective countries based on features such as price, star rating, and number of ratings.

Decision Tree Classifier:

- A Decision Tree is a tree-structured model where:
  - Internal nodes represent tests on features.
  - Branches represent outcomes of the test.
  - Leaf nodes represent class labels.
- Decision Trees split the data using criteria such as Gini Index or Entropy (Information Gain).
- They are easy to interpret but can overfit on noisy data.

Naïve Bayes Classifier:

- Based on Bayes' Theorem with the naïve assumption that features are conditionally independent.
- Formula:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

where:
P(C | X): Posterior probability (probability of class C given data X)
P(X | C): Likelihood
P(C): Prior probability
P(X): Evidence

- Works well on **large datasets**, simple and computationally efficient.
- Performs well even with limited training data.

Conclusion:

- Both Decision Tree and Naïve Bayes classifiers were successfully implemented on the Amazon Bestsellers dataset.
- Decision Tree provided better interpretability and was able to capture complex relationships between features.

- Naïve Bayes was faster and simpler, but its assumption of feature independence might reduce accuracy.
- The experiment demonstrated how real-world data (like Amazon product listings) can be classified into categories such as country, based on product attributes.

Challenges Faced

1. **Data Preprocessing**:
    - The product_price column contained special characters like "₹" and commas, which required cleaning and conversion to numeric values.
    - Some rows had missing values in price, rating, or number of ratings, which had to be handled.
2. **Imbalanced Data**:
   Some countries may have more bestseller records than others, which can bias the classifiers.

Code:

Decision Tree:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay,
classification_report
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("Amazon_bestsellers_items_2025.csv")

# ——— Clean product_price ———
df['product_price'] = (
    df['product_price']
    .astype(str)
    .str.replace("₹", "", regex=False)
    .str.replace("$", "", regex=False)
    .str.replace(",", "", regex=False)
)
df['product_price'] = pd.to_numeric(df['product_price'], errors='coerce')

# Drop rows with missing important values
df = df.dropna(subset=['product_price', 'product_star_rating', 'product_num_ratings',
'country'])

# ——— Select Features ———
X = df[['product_price', 'product_star_rating', 'product_num_ratings']]
y = df['country']
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# ——— Train Decision Tree ———
dt = DecisionTreeClassifier(max_depth=3, random_state=42)  # reduced depth for
readability
dt.fit(X_train, y_train)

# Predictions & Accuracy
y_pred = dt.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred))

# ——— Confusion Matrix ———
cm = confusion_matrix(y_test, y_pred, labels=dt.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=dt.classes_)

plt.figure(figsize=(8, 6))
disp.plot(cmap="Greens", xticks_rotation=45)
plt.title("Decision Tree Confusion Matrix")
plt.show()

# ——— Classification Report ———
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# ——— Visualize Decision Tree ———
plt.figure(figsize=(15, 8))
plot_tree(dt, filled=True, feature_names=X.columns, class_names=dt.classes_,
rounded=True)
plt.title("Decision Tree (max_depth=3)")
plt.show()
```

Naive Bayes:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("Amazon_bestsellers_items_2025.csv")

# ——— Clean product_price ———
df['product_price'] = (
    df['product_price']
```

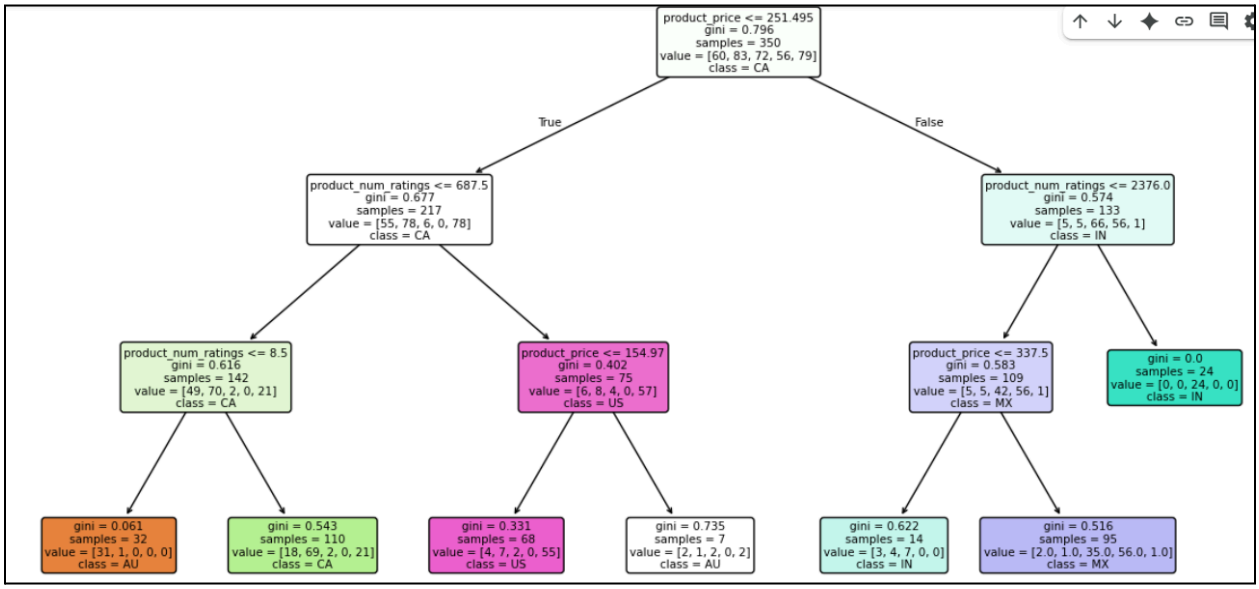
```
    .astype(str)
    .str.replace("₹", "", regex=False)
    .str.replace("$", "", regex=False)
    .str.replace(",", "", regex=False)
)
df['product_price'] = pd.to_numeric(df['product_price'], errors='coerce')

# Drop rows with missing important values
df = df.dropna(subset=['product_price', 'product_star_rating', 'product_num_ratings',
'country'])

# ——— Select Features ———
X = df[['product_price', 'product_star_rating', 'product_num_ratings']]
y = df['country']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Scale numerical features
scaler = StandardScaler()
X_train[X_train.columns] = scaler.fit_transform(X_train)
X_test[X_test.columns] = scaler.transform(X_test)

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

# Accuracy
print("Naive Bayes Accuracy:", nb.score(X_test, y_test))

# ——— Confusion Matrix ———
y_pred = nb.predict(X_test)
cm = confusion_matrix(y_test, y_pred, labels=nb.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=nb.classes_)

plt.figure(figsize=(8, 6))
disp.plot(cmap="Blues", xticks_rotation=45)
plt.title("Naive Bayes Confusion Matrix")
plt.show()

# ——— Classification Report ———
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Output:

Decision Tree:

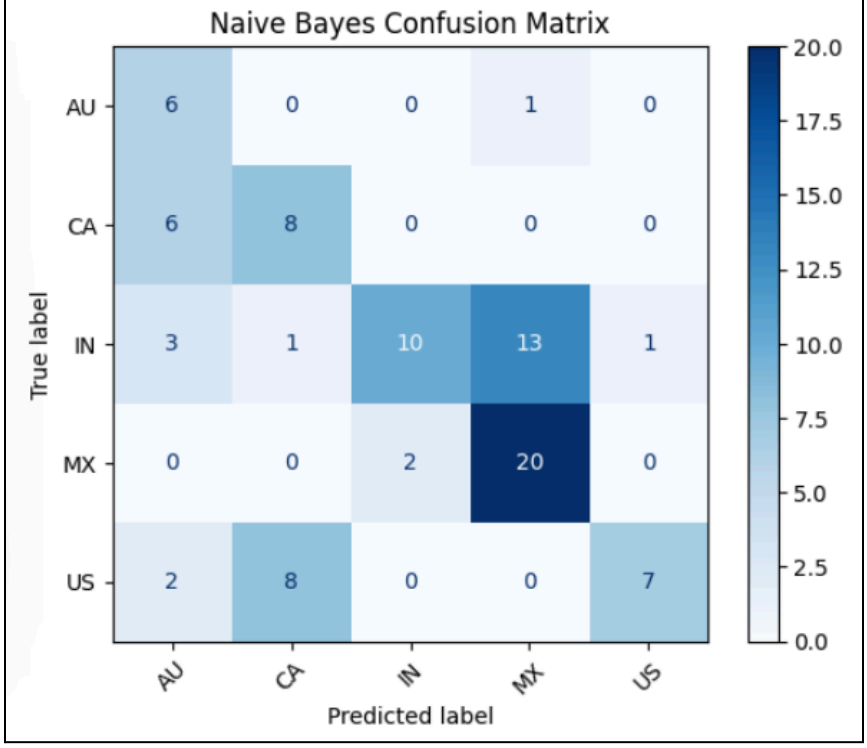
```
Decision Tree Accuracy: 0.6363636363636364
```

```
Classification Report:
              precision    recall  f1-score   support

          AU       0.50      0.57      0.53         7
          CA       0.58      0.79      0.67        14
          IN       1.00      0.29      0.44        28
          MX       0.56      1.00      0.72        22
          US       0.79      0.65      0.71        17

    accuracy                           0.64        88
   macro avg       0.69      0.66      0.62        88
weighted avg       0.74      0.64      0.61        88
```



Naive Bayes Algorithm:

```
Naive Bayes Accuracy: 0.5795454545454546
<Figure size 800x600 with 0 Axes>
```



Classification Report:	precision	recall	f1-score	support
AU	0.35	0.86	0.50	7
CA	0.47	0.57	0.52	14
IN	0.83	0.36	0.50	28
MX	0.59	0.91	0.71	22
US	0.88	0.41	0.56	17
accuracy			0.58	88
macro avg	0.62	0.62	0.56	88
weighted avg	0.68	0.58	0.57	88