Name: Chinmay Chaudhari AIDS-1 D15C/06

DS-Lab Experiment 6

Aim: Classification modelling– Use a classification algorithm and evaluate the performance.

a) Choose classifier for classification problem.
b) Evaluate the performance of classifier.

Perform Classification using (2 of) the below 4 classifiers on the same dataset which you have used
for experiment no 5:
K-Nearest Neighbors (KNN)
Naive Bayes
Support Vector Machines (SVMs)
Decision Tree

Theory:

Decision Tree:
The Decision Tree classifier builds a model by recursively splitting the data based on feature values, creating a tree where each node represents a decision rule and each leaf a class label. This approach is highly interpretable, as the decision rules can be easily visualized and understood.

K-Nearest Neighbors (KNN):
KNN classifies a new instance by finding the k closest training examples based on a distance metric (typically Euclidean distance) and assigning the majority class among these neighbors. It is a non-parametric and intuitive method that performs well when features are properly scaled.

Naive Bayes:
Naive Bayes uses Bayes' theorem with the strong assumption that all features are conditionally independent given the class label. This probabilistic classifier is computationally efficient and performs robustly in high-dimensional settings, despite its simplicity.

Support Vector Machines (SVM):
SVM finds the optimal hyperplane that separates classes by maximizing the margin between them, and it can handle non-linear boundaries through the use of kernel functions. It is especially effective in high-dimensional spaces and tends to offer robust performance with appropriate parameter tuning.

Data Description:

Big Data
14+ columns

age: The age of the individual.
workclass: The type of employment (e.g., private, self-employed, government).
fnlwgt: Final weight, representing the number of people the individual represents.
education: The highest level of education achieved.
education-num: The number of years of education completed.
marital-status: The marital status of the individual (e.g., married, single).
occupation: The type of job or occupation.
relationship: The individual's relationship status within a household (e.g., husband, wife).
race: The race of the individual.
sex: The gender of the individual.
capital-gain: Income from investment sources other than salary/wages.
capital-loss: Losses from investment sources other than salary/wages.
hours-per-week: The number of hours worked per week.
native-country: The country of origin.
income: The income level (<=50K or >50K).

We are selecting decision tree and naiive bayes classification algorithms for classifying the income level of un seen data, based on all the parameters mentioned above.

Implementation

Step 1) Load the Dataset

```
!wget https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data
!wget https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names  # For column names
```

```
--2025-03-18 18:01:32--  https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified
Saving to: ‘adult.data’

adult.data              [ <=>                ]   3.79M  --.-KB/s    in 0.1s

2025-03-18 18:01:33 (34.3 MB/s) - ‘adult.data’ saved [3974305]

--2025-03-18 18:01:33--  https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified
```

Step 2) Preprocess the data

```
import pandas as pd

# Load data
columns = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status',
           'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',
           'hours-per-week', 'native-country', 'income']
df = pd.read_csv('adult.data', names=columns, na_values=' ?', skipinitialspace=True)

# Drop missing values
df = df.dropna()

# Encode target
df['income'] = df['income'].map({'<=50K': 0, '>50K': 1})

# Select features
features = ['age', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week',
            'workclass', 'education', 'marital-status', 'occupation', 'sex']
X = df[features]
y = df['income']

# One-hot encode categorical variables
X = pd.get_dummies(X, columns=['workclass', 'education', 'marital-status', 'occupation', 'sex'], drop_first=True)

print(f"Data shape: {X.shape}, Target shape: {y.shape}")
```

```
Data shape: (32561, 49), Target shape: (32561,)
```

```
[3] df.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	Uni

Missing Values: Real-world data often has gaps. Dropping rows is simple but reduces data (alternatives: imputation).

Target Encoding: Logistic Regression needs a numerical target. We mapped <=50K to 0 and >50K to 1 for binary classification.

One-Hot Encoding: Categorical variables (e.g., occupation) can't be used directly in math-based models. get_dummies converts them to binary columns (e.g., occupation_Exec-managerial: 1 if true, 0 if not). drop_first=True avoids multicollinearity (dummy variable trap).

X is the feature matrix (inputs), y is the target vector (output).

Step 3: Splitting the dataset.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f"Train shape: {X_train.shape}, Test shape: {X_test.shape}")
```

```
Train shape: (26048, 49), Test shape: (6513, 49)
```

Step 4: Training the classifiers..

Train Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

# Initialize and train Decision Tree
dt_clf = DecisionTreeClassifier(max_depth=10, random_state=42)
dt_clf.fit(X_train, y_train)

# Predict
y_pred_dt = dt_clf.predict(X_test)
```

Train Naiive bayes

```
[ ]  from sklearn.naive_bayes import GaussianNB

     # Initialize and train Naive Bayes
     nb_clf = GaussianNB()
     nb_clf.fit(X_train, y_train)

     # Predict
     y_pred_nb = nb_clf.predict(X_test)
```

Step 5: Model Evaluation

Evaluating function:

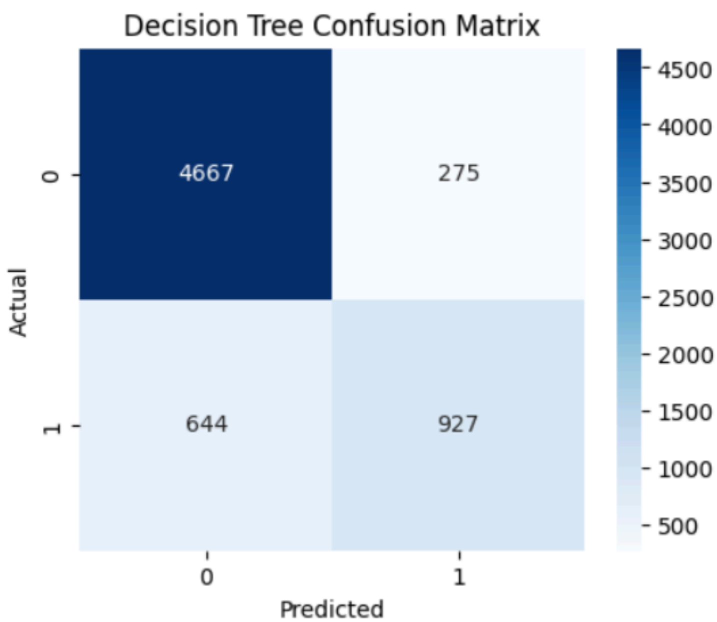
Performance measures

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Function to evaluate and plot
def evaluate_model(y_test, y_pred, model_name):
    print(f"\n{model_name} Performance:")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'{model_name} Confusion Matrix')
    plt.show()
```

```
[ ]  # Evaluate Decision Tree
     evaluate_model(y_test, y_pred_dt, "Decision Tree")
```

```python
# Evaluate Decision Tree
evaluate_model(y_test, y_pred_dt, "Decision Tree")
```

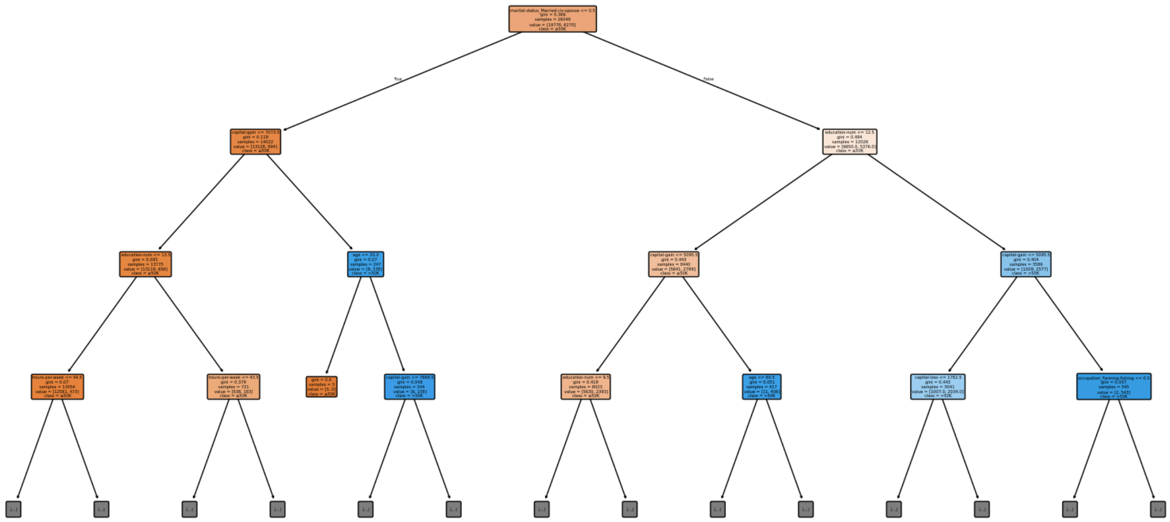
```
Decision Tree Performance:
Accuracy: 0.86

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      4942
           1       0.77      0.59      0.67      1571

    accuracy                           0.86      6513
   macro avg       0.82      0.77      0.79      6513
weighted avg       0.85      0.86      0.85      6513
```



Visualise Tree

```python
from sklearn.tree import DecisionTreeClassifier, plot_tree, export_text
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 10))
plot_tree(dt_clf, feature_names=X.columns, class_names=['≤50K', '>50K'], filled=True, rounded=True, max_depth=3)
plt.title("Decision Tree (Top 3 Levels)")
plt.show()
```

Decision Tree (Top 3 Levels)



Decision rules

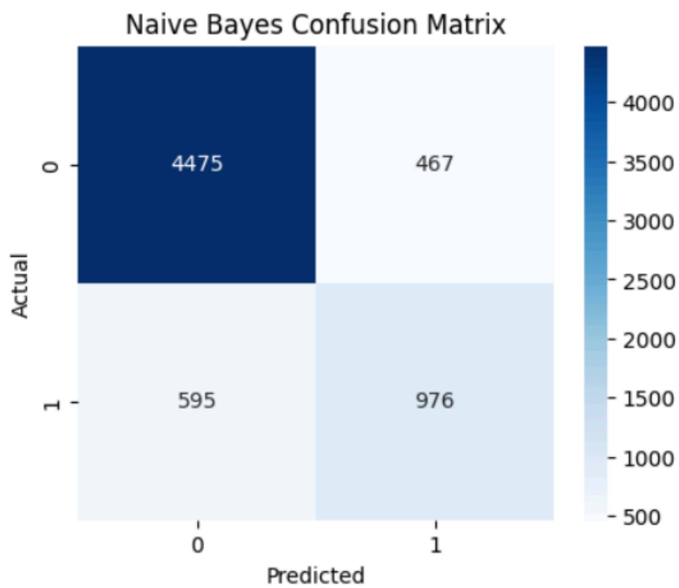
```
[ ]  # Extract Decision Rules
     rules = export_text(dt_clf, feature_names=list(X.columns))
     print("\nDecision Rules:")
     print(rules[:1000])
```

```
Decision Rules:
|--- marital-status_Married-civ-spouse <= 0.50
|   |--- capital-gain <= 7073.50
|   |   |--- education-num <= 13.50
|   |   |   |--- hours-per-week <= 44.50
|   |   |   |   |--- capital-loss <= 2218.50
|   |   |   |   |   |--- age <= 33.50
|   |   |   |   |   |   |--- marital-status_Married-AF-spouse <= 0.50
|   |   |   |   |   |   |   |--- age <= 26.50
|   |   |   |   |   |   |   |   |--- education_5th-6th <= 0.50
|   |   |   |   |   |   |   |   |   |--- occupation_Protective-serv <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- occupation_Protective-serv >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- education_5th-6th >  0.50
|   |   |   |   |   |   |   |   |   |--- workclass_Local-gov <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- workclass_Local-gov >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |
```

Evaluate Naive Bayes

```
evaluate_model(y_test, y_pred_nb, "Naive Bayes")
```

```
Naive Bayes Performance:
Accuracy: 0.84

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.91      0.89      4942
           1       0.68      0.62      0.65      1571

    accuracy                           0.84      6513
   macro avg       0.78      0.76      0.77      6513
weighted avg       0.83      0.84      0.83      6513
```



Conclusion

In Experiment 6, we preprocessed the dataset by encoding categorical features into dummies and splitting it into training and test sets. We tested classifiers, initially facing issues with Naive Bayes due to scaling, then adjusted by using unscaled data. Decision Tree was evaluated with its tree visualization and rules, while Naive Bayes variants were compared for performance. The focus was on selecting a classifier and understanding its fit to our data. Final accuracies: Naive Bayes (0.84), Decision Tree (0.86).