1.Predicting House Prices Using Polynomial Regression

Problem Statement:

A real estate company wants to predict the prices of houses based on the number of bedrooms.

The data they have gathered shows the relationship between the number of bedrooms in a house and its price. However,

the relationship between the number of bedrooms and price is not linear, so a polynomial regression model will be used to better model the data.

Number of Bedrooms (X) vs House Price (Y) in thousands

1 150

2 180

3 210

4 280

5 310

6 400

===============================================================

# Import necessary libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

# Given data: Number of Bedrooms (X) and House Price (Y)

X = np.array([1, 2, 3, 4, 5, 6]).reshape(-1, 1) # Feature: Number of Bedrooms

Y = np.array([150, 180, 210, 280, 310, 400]) # Target: House Price

# Step 1: Visualize the data

plt.scatter(X, Y, color='blue')

plt.title("House Price vs. Number of Bedrooms")

plt.xlabel("Number of Bedrooms")

plt.ylabel("House Price (in thousands)")

plt.show()

# Step 2: Create a Polynomial Feature Transformer

degree = 2 # Polynomial degree 2

poly\_reg = PolynomialFeatures(degree=degree)

X\_poly = poly\_reg.fit\_transform(X)

# Step 3: Create the Polynomial Regression Model and Fit the Data

lin\_reg = LinearRegression()

lin\_reg.fit(X\_poly, Y)

# Step 4: Make Predictions

# Let's predict the house price for 4.5 bedrooms

x\_new = np.array([[4.5]]) # 4.5 bedrooms

x\_new\_poly = poly\_reg.transform(x\_new) # Transform the input to match polynomial degree

y\_pred = lin\_reg.predict(x\_new\_poly)

print(f"Predicted House Price for 4.5 Bedrooms: {y\_pred[0]:.2f} thousand dollars")

# Step 5: Plot the Polynomial Regression Curve

plt.scatter(X, Y, color='blue')

plt.plot(X, lin\_reg.predict(poly\_reg.transform(X)), color='red') # Polynomial regression curve

plt.title("Polynomial Regression: House Price vs. Number of Bedrooms")

plt.xlabel("Number of Bedrooms")

plt.ylabel("House Price (in thousands)")

plt.show()

======================

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix

from sklearn.preprocessing import LabelEncoder

# Create synthetic dataset

data = {

'applicant\_income': [45000, 54000, 120000, 80000, 60000, 70000, 85000, 90000, 35000, 100000],

'credit\_score': [650, 700, 750, 800, 680, 690, 710, 730, 620, 760],

'loan\_amount': [10000, 12000, 30000, 25000, 15000, 20000, 22000, 24000, 8000, 28000],

'employment\_status': ['Employed', 'Employed', 'Self-Employed', 'Employed', 'Self-Employed',

'Employed', 'Self-Employed', 'Employed', 'Employed', 'Self-Employed'],

'loan\_approved': ['Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No'] # Target variable

}

df = pd.DataFrame(data)

# Encode categorical data

label\_encoder = LabelEncoder()

df['employment\_status'] = label\_encoder.fit\_transform(df['employment\_status'])

df['loan\_approved'] = label\_encoder.fit\_transform(df['loan\_approved'])

# Features and target variable

X = df.drop(columns=['loan\_approved'])

y = df['loan\_approved']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Decision Tree Classifier

dt = DecisionTreeClassifier(random\_state=42)

dt.fit(X\_train, y\_train)

# Make predictions

y\_pred = dt.predict(X\_test)

# Evaluate the model

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.2f}")

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

# Plot confusion matrix

plt.figure(figsize=(6, 6))

sns.heatmap(confusion\_matrix(y\_test, y\_pred), annot=True, fmt="d", cmap="Blues", xticklabels=["No", "Yes"], yticklabels=["No", "Yes"])

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

# Visualize the Decision Tree

plt.figure(figsize=(12, 8))

plot\_tree(dt, filled=True, feature\_names=X.columns, class\_names=['No', 'Yes'], fontsize=10)

plt.title('Decision Tree Classifier')

plt.show()

========================================