TASK 4

SALES PREDICTION USING PYTHON

- Sales prediction involves forecasting the amount of a product that customers will purchase, taking into account various factors such as advertising expenditure, target audience segmentation, and advertising platform selection.
- In businesses that offer products or services, the role of a Data Scientist is crucial for predicting future sales. They utilize machine learning techniques in Python to analyze and interpret data, allowing them to make informed decisions regarding advertising costs. By leveraging these predictions, businesses can optimize their advertising strategies and maximize sales potential. Let's embark on the journey of sales prediction using machine learning in Python.

DATASET CLICK HERE

DATASET CLICK HERE

(use any one dataset)

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import tkinter as tk
from tkinter import filedialog

# Function to load the dataset
def load_file():
    root = tk.Tk()
    root.withdraw()  # Hide the root window
    file_path = filedialog.askopenfilename(title="Select your dataset", filetypes=(("CSV files", "*.csv"), ("All files", "*.*")))
    if file_path:
        return pd.read_csv(file_path)
    else:
        print("No file selected.")
        return None

# Function for data preprocessing
def preprocess_data(df):
    # Check for missing values and fill them (or drop if necessary)
    df.fillna(df.mean(), inplace=True)

    # If there are categorical variables, apply one-hot encoding
    df = pd.get_dummies(df, drop_first=True)

    return df

# Function to train the model and evaluate it
def train_and_evaluate_model(X_train, X_test, y_train, y_test):
    # Initialize the Linear Regression model
    model = LinearRegression()

    # Train the model
    model.fit(X_train, y_train)

    # Make predictions on the test data
    y_pred = model.predict(X_test)

    # Calculate performance metrics
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
```

```python
    r2 = r2_score(y_test, y_pred)

    print(f'Mean Absolute Error (MAE): {mae}')
    print(f'Mean Squared Error (MSE): {mse}')
    print(f'R-squared: {r2}')

    # Plotting actual vs predicted sales
    plt.scatter(y_test, y_pred)
    plt.xlabel('Actual Sales')
    plt.ylabel('Predicted Sales')
    plt.title('Actual vs Predicted Sales')
    plt.show()

# Main function to run the process
def main():
    print("Welcome to the Sales Prediction System!")

    # Let the user choose the file
    df = load_file()
    if df is None:
        return

    # Show the first few rows of the dataset
    print("Loaded dataset:")
    print(df.head())

    # Preprocess the data
    df = preprocess_data(df)

    # Check the structure of the data
    print("Preprocessed data:")
    print(df.head())

    # Split data into features and target variable
    if 'Sales' not in df.columns:
        print("The dataset must contain a 'Sales' column. Please ensure your dataset has the appropriate target column.")
        return

    X = df.drop('Sales', axis=1)  # Features
    y = df['Sales']               # Target

    # Split data into training and testing sets (80% training, 20% testing)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train the model and evaluate performance
    train_and_evaluate_model(X_train, X_test, y_train, y_test)
```

```python
# Main function to run the process
def main():
    print("Welcome to the Sales Prediction System!")

    # Let the user choose the file
    df = load_file()
    if df is None:
        return

    # Show the first few rows of the dataset
    print("Loaded dataset:")
    print(df.head())

    # Preprocess the data
    df = preprocess_data(df)

    # Check the structure of the data
    print("Preprocessed data:")
    print(df.head())

    # Split data into features and target variable
    if 'Sales' not in df.columns:
        print("The dataset must contain a 'Sales' column. Please ensure your dataset has the appropriate target column.")
        return

    X = df.drop('Sales', axis=1)  # Features
    y = df['Sales']               # Target

    # Split data into training and testing sets (80% training, 20% testing)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train the model and evaluate performance
    train_and_evaluate_model(X_train, X_test, y_train, y_test)

if __name__ == "__main__":
    main()
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:\Users\HP\OneDrive\Desktop\CA\task 4.py ==============
Welcome to the Sales Prediction System!
Loaded dataset:
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
Preprocessed data:
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
Mean Absolute Error (MAE): 1.2748262109549338
Mean Squared Error (MSE): 2.9077569102710896
R-squared: 0.9059011844150826
```