**Artificial Intelligence**



Session: 2022 – 2026

**Submitted by:**

Abdur Rehman Kazim 2022-CS-115

**Supervised by:**

Sir Noman Shafi

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

# **Task 1**

## **Code Fence:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, MinMaxScaler

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

# Setting random seed for reproducibility

np.random.seed(0)

n = 1000

# Creating synthetic customer data

customer\_id = np.arange(1, n + 1)

age = np.random.randint(18, 70, n)

annual\_income = np.random.randint(20000, 120000, n)

gender = np.random.choice(['Male', 'Female'], n)

purchased = np.random.choice([0, 1], n)

# Creating a DataFrame

df = pd.DataFrame({

'CustomerID': customer\_id,

'Age': age,

'Annual Income': annual\_income,

'Gender': gender,

'Purchased': purchased

})

df.to\_csv('customer\_data.csv', index=False)

print(df.head(10))

# Checking for missing values

print(df.isnull().sum())

# Encoding categorical variables

label\_encoder = LabelEncoder()

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

print(df.head(10))

# Feature Scaling

scaler = MinMaxScaler()

df[['Age', 'Annual Income']] = scaler.fit\_transform(df[['Age', 'Annual Income']])

print(df.head(10))

# Data Visualization

plt.hist(df['Age'], bins=10, color='blue', edgecolor='black')

plt.title('Age Distribution')

plt.xlabel('Age')

plt.ylabel('Count')

plt.show()

plt.scatter(df['Age'], df['Annual Income'], color='red')

plt.title('Age vs Annual Income')

plt.xlabel('Age')

plt.ylabel('Annual Income')

plt.show()

# Correlation Analysis

correlation\_matrix = df[['Age', 'Annual Income', 'Purchased']].corr()

print(correlation\_matrix)

# Feature Engineering

df['Income per Age'] = df['Annual Income'] / df['Age']

print(df.head(10))

# Preparing Data for Modeling

df.drop('CustomerID', axis=1, inplace=True)

print(df.head(10))

# Splitting the data into training and testing sets

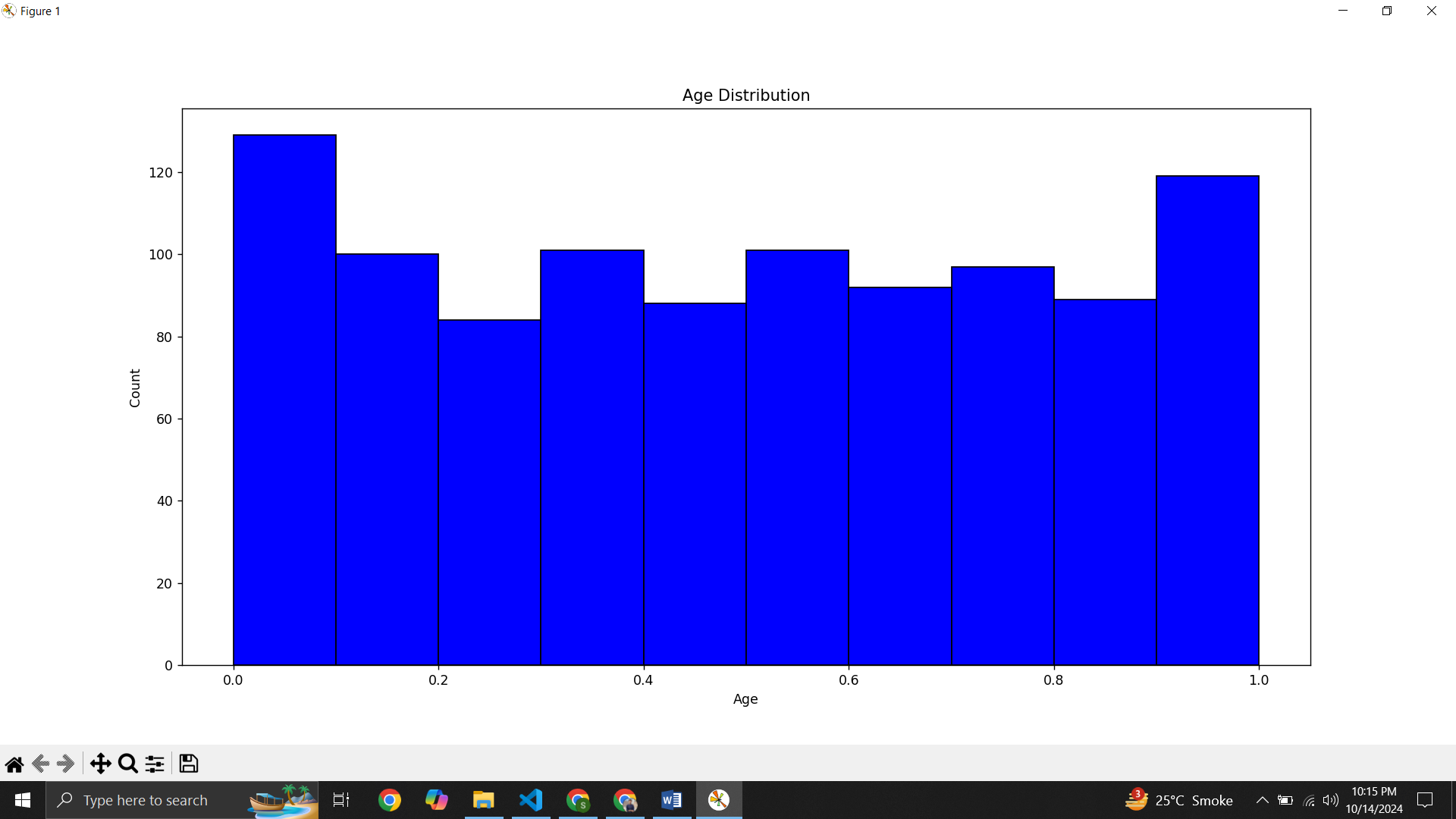
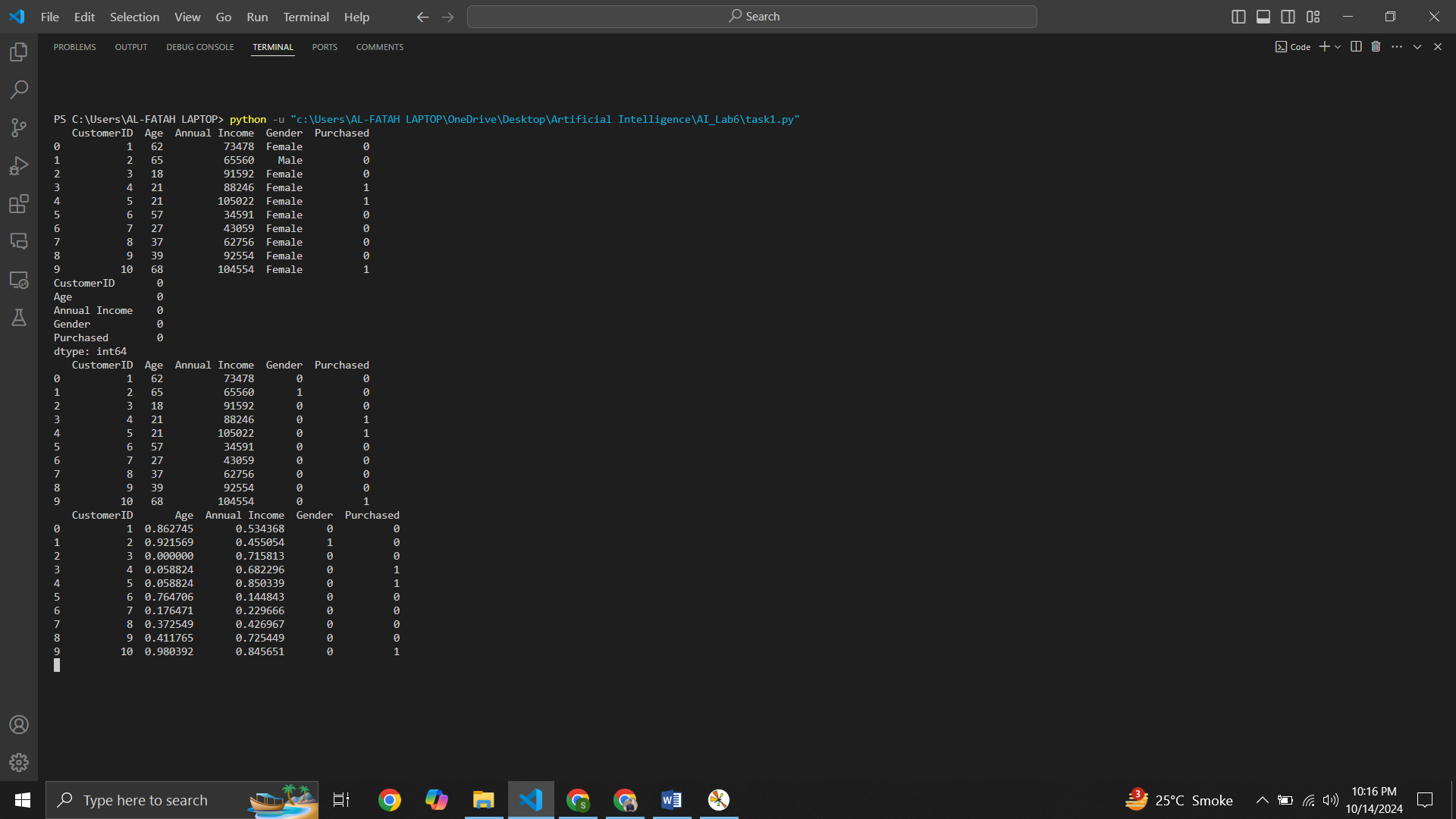
X = df.drop('Purchased', axis=1)

y = df['Purchased']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

## **Output**



# **Task 2**

## **Code Fence:**

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

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder, StandardScaler

import matplotlib.pyplot as plt

# Setting random seed for reproducibility

np.random.seed(0)

# Generate Dummy Data

n = 1500

employee\_id = np.arange(1, n + 1)

age = np.random.randint(22, 61, n)

years\_of\_experience = np.random.randint(1, 50, n) # Allow values over 40 for outliers

gender = np.random.choice(['Male', 'Female'], n)

performance\_rating = np.random.randint(1, 6, n)

# Create DataFrame

df = pd.DataFrame({

'EmployeeID': employee\_id,

'Age': age,

'Years of Experience': years\_of\_experience,

'Gender': gender,

'Performance Rating': performance\_rating

})

# Save to CSV and print first 10 rows

df.to\_csv('employee\_performance\_data.csv', index=False)

print(df.head(10))

# Check for missing values

print(df.isnull().sum())

# Encode categorical variables

label\_encoder = LabelEncoder()

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

print(df.head(10))

# Detecting outliers using box plot

plt.boxplot(df['Years of Experience'])

plt.title('Years of Experience Box Plot')

plt.show()

# Remove outliers (values over 40)

df = df[df['Years of Experience'] <= 40]

print(df.head(10))

# Scale Age and Years of Experience columns

scaler = StandardScaler()

df[['Age', 'Years of Experience']] = scaler.fit\_transform(df[['Age', 'Years of Experience']])

print(df.head(10))

# Data Visualization

# Box plot for Performance Rating

plt.boxplot(df['Performance Rating'])

plt.title('Performance Rating Box Plot')

plt.show()

# Scatter plot: Years of Experience vs Performance Rating

plt.scatter(df['Years of Experience'], df['Performance Rating'])

plt.title('Years of Experience vs Performance Rating')

plt.xlabel('Years of Experience')

plt.ylabel('Performance Rating')

plt.show()

# Correlation analysis

correlation\_matrix = df[['Age', 'Years of Experience', 'Performance Rating']].corr()

print(correlation\_matrix)

# Create a new feature: Experience per Age

df['Experience per Age'] = df['Years of Experience'] / df['Age']

print(df.head(10))

# Drop irrelevant columns

df = df.drop(columns=['EmployeeID'])

print(df.head(10))

# Split the data into training and testing sets

X = df.drop(columns=['Performance Rating'])

y = df['Performance Rating']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

## **Output**

