**CS371L – Artificial Intelligence**

**AI Lab 6 Report**



Session: 2022 – 2026

**Submitted by:**

Muhammad Tayyab 2022-CS-133

**Submitted to:**

Mr. Nauman Shafee

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

# Case Study 1

## Task 1:

## customer\_id = np.arange(1, 1001)

## age = np.random.randint(18, 71, size=1000)

## annual\_income = np.random.randint(20000, 120001, size=1000)

## gender = np.random.choice(['Male', 'Female'], size=1000)

## purchased = np.random.choice([0, 1], size=1000)

## data = pd.DataFrame({

## 'CustomerID': customer\_id,

## 'Age': age,

## 'Annual Income': annual\_income,

## 'Gender': gender,

## 'Purchased': purchased

## })

## *# Display the first few rows of the dataset*

## *print(data.head())*

## Task 2:

## *# Check for missing values*

## missing\_values = data.isnull().sum()

## *print(missing\_values)*

## Task 3:

## data.loc[np.random.choice(data.index, size=50, replace=False), 'Annual Income'] = np.nan

## *print(data['Annual Income'].isnull().sum())*

median\_income = data['Annual Income'].median()

data['Annual Income'].fillna(median\_income, inplace=True)

*print(data['Annual Income'].isnull().sum())*

## Task 4:

## data['Gender'] = data['Gender'].map({'Male': 0, 'Female': 1})

## *print(data.head())*

## Task 5:

scaler = MinMaxScaler()

*# Scale Age and Annual Income columns*

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

*print(data[['Age', 'Annual Income']].head())*

## Task 6:

*# Create a histogram*

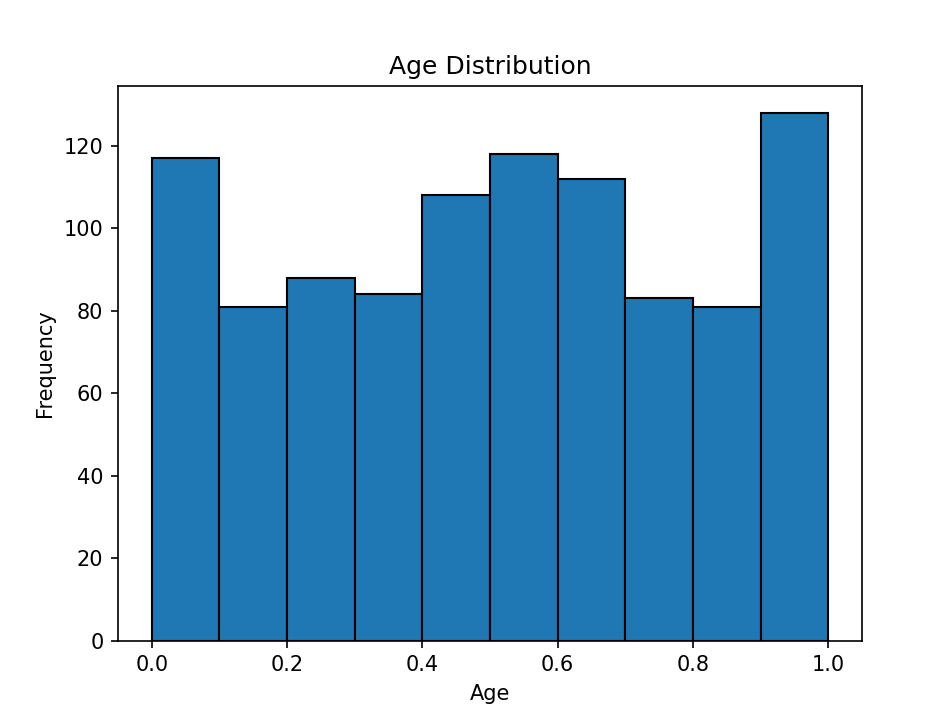
plt.hist(data['Age'], bins=10, edgecolor='black')

plt.title('Age Distribution')

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.show()



*# Create a scatter plot*

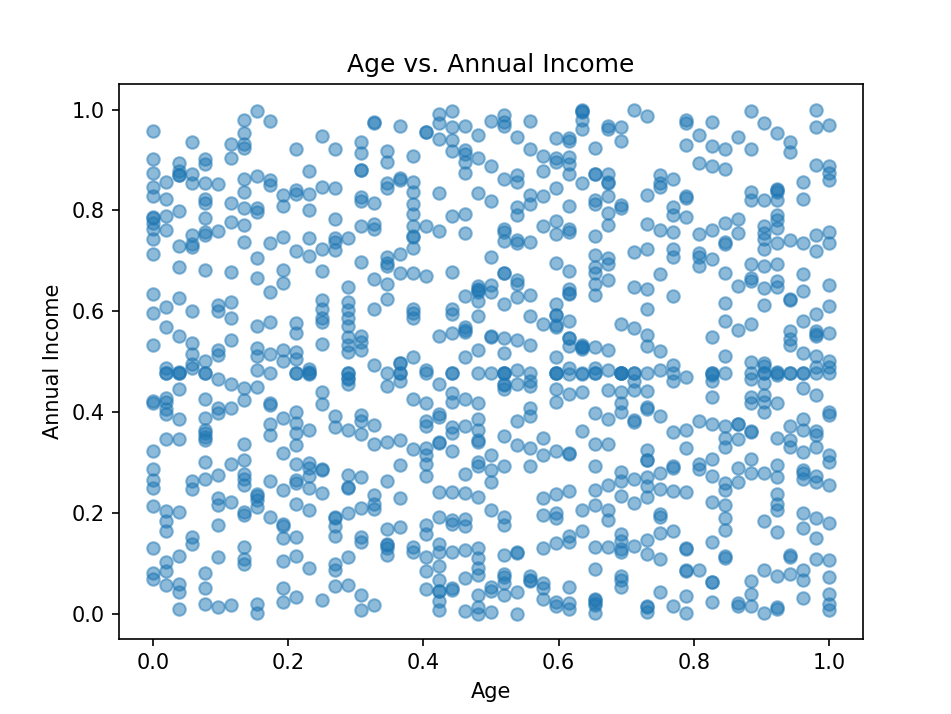
plt.scatter(data['Age'], data['Annual Income'], alpha=0.5)

plt.title('Age vs. Annual Income')

plt.xlabel('Age')

plt.ylabel('Annual Income')

plt.show()



## Task 7:

*# Calculate the correlation matrix*

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

*print(correlation\_matrix)*

## Task 8:

*# data['Income per Age'] = data['Annual Income'] / data['Age']*

*print(data[['Age', 'Annual Income', 'Income per Age']].head())*

## Task 9:

data = data.drop('CustomerID', axis=1)

print(data.head())

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

y = data['Purchased']

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

print("Training set shape:", X\_train.shape, ",", y\_train.shape)

print("Testing set shape:", X\_test.shape, ",", y\_test.shape)

# Case Study 2

## Task 1:

## Employee\_id = np.arange(1, 1501)

## Age = np.random.randint(20, 60, size=1500)

## Experience = np.random.randint(1, 40, size=1500)

## Gender = np.random.choice(["Male", "Female"], size=1500)

## PerformanceRating = np.random.randint(1, 5, size=1500)

## data\_table = pd.DataFrame({

## "Employee ID": Employee\_id,

## "Employee Age": Age,

## "Experience": Experience,

## "Gender": Gender,

## "Performance Rating": PerformanceRating

## })

Task 2:

## *# Check for missing values*

## missing\_values = data.isnull().sum()

## *print(missing\_values)*

## Task 3:

## data\_table.loc[np.random.choice(data\_table.index, size=50, replace=False), 'Experience'] = np.nan

## data\_table['Experience'].fillna(data\_table['Experience'].median(), inplace=True)

## Task 4:

## data\_table['Gender'] = data\_table['Gender'].map({"Male": 0, "Female": 1})

## Task 5:

## plt.figure(figsize=(8, 6))

## plt.boxplot(data\_table['Experience'], vert=False, patch\_artist=True, boxprops=dict(facecolor='lightgreen'))

## plt.title('Boxplot for Years of Experience')

## plt.xlabel('Years of Experience')

## plt.show()

## *# IQR for outlier detection*

## Q1 = data\_table['Experience'].quantile(0.25)

## Q3 = data\_table['Experience'].quantile(0.75)

## IQR = Q3 - Q1

## lower\_bound = Q1 - 1.5 \* IQR

## upper\_bound = Q3 + 1.5 \* IQR

## print("Lower Bound:", lower\_bound, " Upper Bound:", upper\_bound)

## *# Option 1:*

## data\_cleaned = data\_table[(data\_table['Experience'] >= lower\_bound) & (data\_table['Experience'] <= upper\_bound)]

## *# Option 2: CAp Outliers*

## data\_table['Experience'] = np.where(data\_table['Experience'] > 40, 40, data\_table['Experience'])

A green rectangular object with black lines

Description automatically generated

## Task 6:

## scaler = StandardScaler()

## columns\_to\_scale = ['Employee Age', 'Experience']

## data\_scaled = data\_table.copy()

## data\_table[columns\_to\_scale] = scaler.fit\_transform(data\_table[columns\_to\_scale])

## print(data\_scaled.head())

## Task 7:

# plt.figure(figsize=(8, 6))

# plt.boxplot(data\_table['Performance Rating'], vert=False, patch\_artist=True, boxprops=dict(facecolor='lightblue'))

# plt.title('Box Plot of Performance Rating')

# plt.xlabel('Performance Rating')

# plt.show()

A blue rectangular object with black lines

Description automatically generated

# *# Scatter Plot*

# plt.figure(figsize=(8, 6))

# plt.scatter(data\_table['Experience'], data\_table['Performance Rating'], color='green')

# plt.title('Scatter Plot: Years of Experience vs. Performance Rating')

# plt.xlabel('Years of Experience')

# plt.ylabel('Performance Rating')

# plt.grid(True)

# plt.show()

A graph with green dots

Description automatically generated

## Task 8:

correlation\_matrix = data\_table[['Age', 'Experience', 'PerformanceRating']].corr()

Task 9:

data\_table['Experience Per Age'] = data\_table['Experience'] / data\_table['Age']

*print(data\_table[['Age', 'Annual Income', 'Income per Age']].head())*

Task 10:

data = data\_table.drop('Employee ID', axis=1)

print(data.head())

X = data\_table.drop('Employee ID', axis=1)

y = data\_table['Employee ID']

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

print("Training set shape:", X\_train.shape, ",", y\_train.shape)

print("Testing set shape:", X\_test.shape, ",", y\_test.shape)