ICP – 4

Venkata Suraj, Gamini
700744962

GitHub: https://github.com/SurajGamini18/Neural-Networks-Deep-Learning-Assignments

Video Link: https://drive.google.com/file/d/11QQHx11gvbSUD1YaCACkKRa0pGilH6su/view?usp=sharing

Q1:

Code:

```
import pandas as pd

# Load the CSV file into a DataFrame
df = pd.read_csv('/content/data.csv')

# Display the basic statistical description about the data
basic_stats = df.describe()

# Check if the data has null values
null_values = df.isnull().sum()

# Replace the null values with the mean of their respective column
df_filled = df.fillna(df.mean())

# Aggregating data for at least two columns: min, max, count, mean
selected_columns = df_filled[['Calories', 'Pulse']]  # Example: selecting 'Calories' and 'Pulse'
aggregated_data = selected_columns.agg(['min', 'max', 'count', 'mean'])

# Filter the dataframe for calories values between 500 and 1000
filtered_500_1000 = df_filled[(df_filled['Calories'] >= 500) & (df_filled['Calories'] <= 1000)]

# Filter the dataframe for calories values > 500 and pulse < 100
filtered_calories_pulse = df_filled[(df_filled['Calories'] > 500) & (df_filled['Pulse'] < 100)]

# Creating a new "df_modified" dataframe without the "Maxpulse" column
df_modified = df_filled.drop(columns=['Maxpulse'])

# Deleting the "Maxpulse" column from the main df dataframe
df.drop(columns=['Maxpulse'], inplace=True)

# Convert the datatype of Calories column to int
df['Calories'] = pd.to_numeric(df['Calories'], downcast='integer', errors='coerce')
df['Calories'].fillna(df['Calories'].mean(), inplace=True)  # Re-fill if any NaN introduced by coercion


basic_stats, null_values, aggregated_data, filtered_500_1000.shape, filtered_calories_pulse.shape, df_modified.head(), df.head()

import matplotlib.pyplot as plt

# Creating a scatter plot for the Duration and Calories columns
plt.figure(figsize=(10, 6))
plt.scatter(df['Duration'], df['Calories'], color='blue', alpha=0.5)
plt.title('Scatter Plot of Duration vs. Calories')
plt.xlabel('Duration (in minutes)')
plt.ylabel('Calories Burned')
plt.grid(True)
plt.show()
```

Explanation:

1. Data Loading and Preparation: Reads a dataset from a CSV file into a DataFrame and checks for null values.

2. Statistical Analysis: Generates basic statistical descriptions of the dataset, such as mean, standard deviation, minimum, and maximum values.

3. Handling Missing Data: Replaces any missing values in the dataset with the mean of their respective columns.

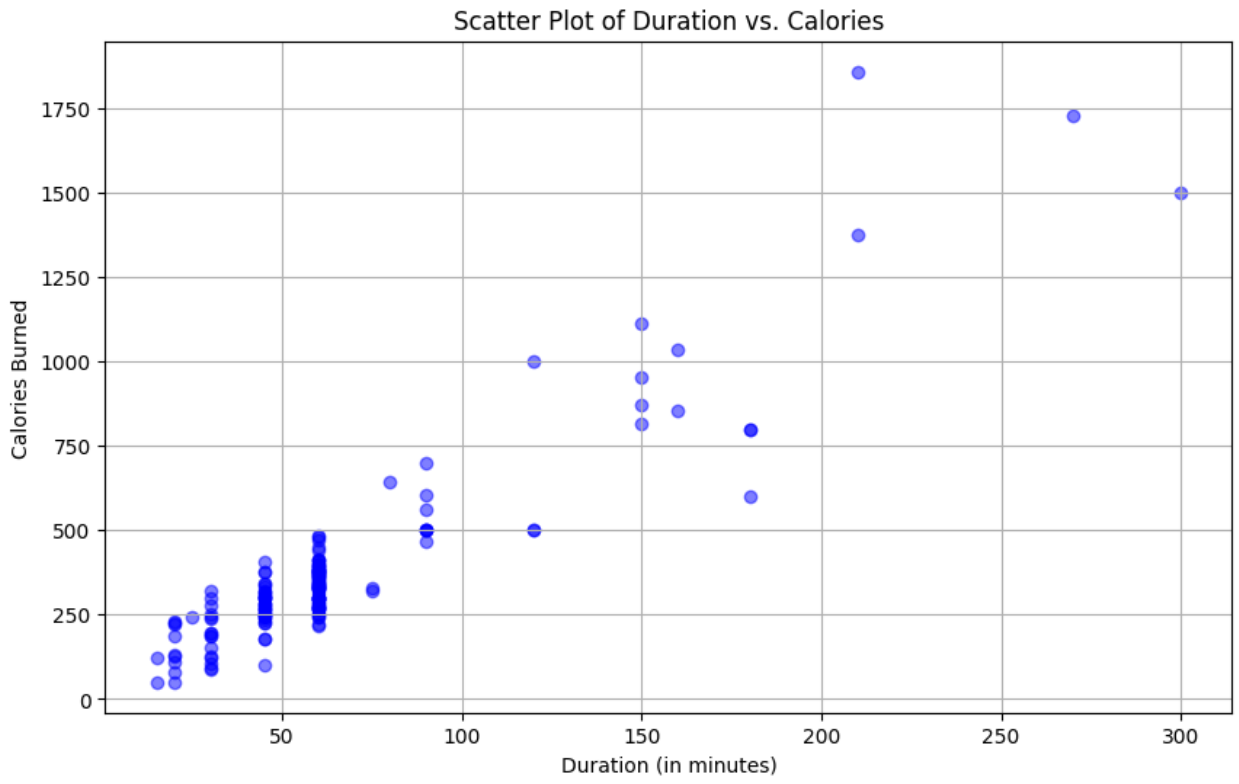
4. Data Aggregation: Aggregates data for 'Calories' and 'Pulse' columns to find their minimum, maximum, average, and count.

5. Data Filtering: Creates subsets of the data based on specific criteria related to 'Calories' and 'Pulse' values.

6. DataFrame Modification: Removes a specific column ('Maxpulse') from the dataset and adjusts the 'Calories' column to integer data type.

7. Data Visualization: Constructs a scatter plot to visually represent the relationship between 'Duration' and 'Calories' in the data.

Output:



Q2:

Code:

```
[ ]  import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
     from sklearn.metrics import mean_squared_error

     # Load the dataset
     salary_data = pd.read_csv('/content/Salary_Data (2).csv')  # Replace with your file path

     # Splitting the dataset into training and testing sets (1/3 for testing)
     X = salary_data.iloc[:, :-1].values  # Features (Years of Experience)
     y = salary_data.iloc[:, -1].values   # Target (Salary)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0)

     # Training the Linear Regression model
     model = LinearRegression()
     model.fit(X_train, y_train)

     # Predicting the Test set results
     y_pred = model.predict(X_test)

     # Calculating the Mean Squared Error
     mse = mean_squared_error(y_test, y_pred)
     print(f"Mean Squared Error: {mse}")

     # Visualizing the Training set results
     plt.figure(figsize=(14, 7))

     plt.subplot(1, 2, 1)
     plt.scatter(X_train, y_train, color='red', label='Actual')
     plt.plot(X_train, model.predict(X_train), color='blue', label='Predicted')
     plt.title('Salary vs. Experience (Training set)')
     plt.xlabel('Years of Experience')
     plt.ylabel('Salary')
     plt.legend()

     # Visualizing the Test set results
     plt.subplot(1, 2, 2)
     plt.scatter(X_test, y_test, color='red', label='Actual')
     plt.plot(X_train, model.predict(X_train), color='blue', label='Model')
     plt.title('Salary vs. Experience (Test set)')
     plt.xlabel('Years of Experience')
     plt.ylabel('Salary')
     plt.legend()

     plt.tight_layout()
     plt.show()
```

Explanation:

1. Imports Necessary Libraries: Utilizes pandas for data handling, matplotlib for visualization, and sklearn for machine learning tasks.

2. Loads and Prepares Data: Reads a salary dataset into a DataFrame and splits it into features (years of experience) and target (salary).

3. Splits Data into Train and Test Sets: Divides the data into training and testing subsets, with one-third of the data reserved for testing.

4. Trains a Linear Regression Model: Fits a linear regression model to the training data to understand the relationship between experience and salary.

5. Predicts Salaries and Evaluates Model: Uses the model to predict salaries on the test set and calculates the mean squared error to assess the model's performance.

6. Visualizes Results: Creates scatter plots to visually compare actual vs. predicted salaries for both training and testing data.

Output: