**Neural Networks & Deep Learning - ICP-4**

**Name**: Abrar Pasha Mohammed

#**700 No**: 700744984

GitHub Link: <https://github.com/Abrar2456/Neural-Networks-ICP4.git>

Video Link: <https://drive.google.com/file/d/15F27aWkXNsCSM-eOEr6lk6wxYA0dLR30/view?usp=sharing>

# TASK 1:

1. Handling Data:

a) Utilizes pd.read\_csv('data.csv') to import a CSV file into a pandas DataFrame (df).

b) Employs df.describe() to present statistical summaries for the DataFrame.

c) Assesses missing values in each column using df.isnull().sum().

2. Addressing Missing Data:

a) Populates missing values in the 'Calories' column with the mean of that column using fillna and inplace=True.

3. Grouping and Aggregating:

a) Groups the DataFrame by 'Duration' and 'Pulse' columns, applying aggregation functions to 'Calories' and 'Maxpulse' columns within each group.

4. Data Filtering:

a) Selects rows where 'Calories' fall within the range of 500 to 1000.

b) Filters rows where 'Calories' surpass 500 and 'Pulse' is below 100.

6. DataFrame Manipulation:

a) Constructs a modified DataFrame (df\_modified) by excluding the 'Maxpulse' column.

b) Attempts to remove the 'Maxpulse' column from the original DataFrame (df) without modifying it in-place.

7. Data Type Conversion and Visualization:

a) Converts the 'Calories' column to an integer data type.

b) Verifies and prints the data type of the first element in the 'Calories' column.

c) Generates a scatter plot using 'Duration' on the x-axis and 'Calories' on the y-axis.

A graph with blue dots

Description automatically generated

# TASK 2 :

1. Data Loading and Exploration:

a) Imports a CSV file ('Salary\_Data.csv') into a pandas DataFrame (sdf) using pd.read\_csv('Salary\_Data.csv').

b) Utilizes sdf.describe() to present statistical summaries of the DataFrame.

2. Data Splitting for Training and Testing:

a) Imports the train\_test\_split function from scikit-learn with from sklearn.model\_selection import train\_test\_split.

b) Divides the dataset into training and testing sets, allocating 67% for training and 33% for testing using x\_train, x\_test, y\_train, y\_test = train\_test\_split(...).

3. Linear Regression Model Initialization and Training:

a) Imports the Linear Regression model from scikit-learn with from sklearn.linear\_model import LinearRegression.

b) Initializes a linear regression model with m = LinearRegression().

c) Trains the linear regression model on the training data using m.fit(x\_train, y\_train).

4. Predictions and Evaluation:

a) Uses the trained model to make predictions on the test data with y\_pred = m.predict(x\_test).

b) Imports the mean\_squared\_error function from scikit-learn as ms with from sklearn.metrics import mean\_squared\_error as ms.

c) Computes the mean squared error between predicted and actual values in the test set using ms(y\_pred, y\_test).

5. Visualization:

a) Imports the Matplotlib library for data visualization with import matplotlib.pyplot as plt.

b) Creates a scatter plot of the training data using plt.scatter(x\_train, y\_train).

c) Adds the test data points to the scatter plot with plt.scatter(x\_test, y\_test).

Overall Workflow:

The code follows a conventional machine learning workflow, encompassing steps such as data loading, splitting, model training, prediction, evaluation, and visualization, all centered around a simple linear regression model.

A graph with blue dots

Description automatically generated

A graph with blue dots

Description automatically generated

**THE END**