

EXPERIMENT 1

Aim: Implement Linear and Logistic Regression on real-world datasets

1.Dataset Source: Custom dataset : [Link](#)

2. Dataset Description

The dataset consists of step-count data collected from a wearable fitness device (Fitbit/Dafit) and exported as CSV files. The raw data contains daily records with the following fields:

- timestamp – Date and time of recorded steps
- steps – Number of steps taken
- data source – Source of the recorded data

During preprocessing, daily step data was aggregated into **monthly average steps**.

Features:

- **MonthIndex** (numerical representation of month: 1, 2, 3, ...)

Target Variable:

- **Monthly Average Steps**

Dataset Characteristics:

- Time-series dataset
- Small sample size (number of months available)
- Shows trend variation (increase followed by decrease)

3. Mathematical Formulation of the Algorithm

The model used is **Simple Linear Regression**.

Model Equation:

$$y = \beta_0 + \beta_1 x$$

Where:

- y = Predicted monthly average steps
- x = Month index
- β_0 = Intercept
- β_1 = Slope

The model parameters are estimated using the **Least Squares Method**, which minimizes the Mean Squared Error (MSE).

Mean Squared Error (MSE):

$$\text{MSE} = (1 / n) \times \sum (y_i - \hat{y}_i)^2$$

for $i = 1$ to n

Where:

- n = Number of observations
- y_i = Actual value
- \hat{y}_i = Predicted value

4. Algorithm Limitations

1. Assumes linear relationship between variables.
2. Cannot capture non-linear or seasonal patterns.
3. Sensitive to outliers.
4. Performance decreases with very small datasets.
5. Long-term future predictions may be unreliable.

5. Methodology / Workflow

1. Data Export – CSV files obtained from fitness app.
2. Data Cleaning – Converted timestamp to datetime format.
3. Feature Engineering – Extracted month and computed monthly averages.

4. Model Training – Applied Linear Regression using MonthIndex as input.
5. Prediction – Predicted next month's average steps.
6. Visualization – Plotted actual values and regression line.
7. Evaluation – Computed R^2 score and error metrics.

Workflow summary:

Data Collection → Preprocessing → Monthly Aggregation
→ Model Training → Prediction → Evaluation

6. Performance Analysis

The model was evaluated using:

- **R^2 Score** – Measures how well the model explains variance in the data.
- **Mean Squared Error (MSE)** – Measures average squared prediction error.

Due to limited data, the model captures only the overall trend and may underfit if the data contains non-linear patterns.

7. Hyperparameter Tuning

Linear Regression has limited hyperparameters:

- `fit_intercept` (True/False)

Tuning impact is minimal for simple regression.

If Polynomial Regression is applied:

- degree becomes the main hyperparameter.
- Higher degree improves flexibility but increases risk of overfitting.

Code:

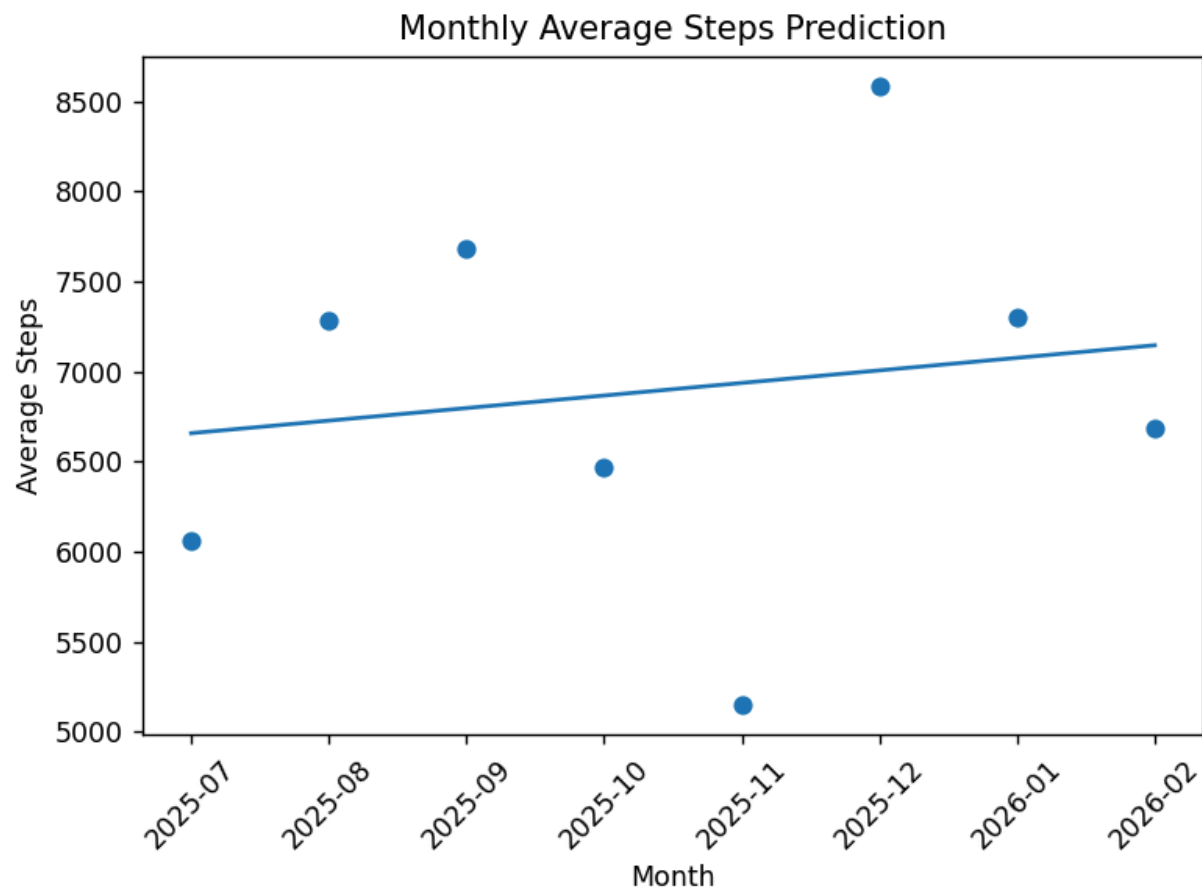
```
1 import pandas as pd
2 import os
3 from sklearn.linear_model import LinearRegression
4 import matplotlib.pyplot as plt
5
6
7 folder_path = "."
8 csv_files = [f for f in os.listdir(folder_path) if f.endswith(".csv")]
9
10 df_list = []
11
12 for file in csv_files:
13     df = pd.read_csv(file)
14     df_list.append(df)
15
16 data = pd.concat(df_list, ignore_index=True)
17
18 print("Files loaded:", csv_files)
19 print("Columns:", data.columns)
20
21
22
23 data['timestamp'] = pd.to_datetime(data['timestamp'])
24 data = data.sort_values('timestamp')
25
26
27
28 data['Date'] = data['timestamp'].dt.date
29 daily_steps = data.groupby('Date')['steps'].sum().reset_index()
30
31
32
33 daily_steps['Date'] = pd.to_datetime(daily_steps['Date'])
34 daily_steps['YearMonth'] = daily_steps['Date'].dt.to_period('M')
35
36 monthly_avg = daily_steps.groupby('YearMonth')['steps'].mean().reset_index()
```

```

38 print("\nMonthly Average Steps:")
39 print(monthly_avg)
40
41
42
43 monthly_avg['MonthLabel'] = monthly_avg['YearMonth'].astype(str)
44 monthly_avg['MonthIndex'] = range(len(monthly_avg))
45
46 X = monthly_avg[['MonthIndex']]
47 y = monthly_avg['steps']
48
49
50 model = LinearRegression()
51 model.fit(X, y)
52
53 print("\nIntercept:", model.intercept_)
54 print("Slope:", model.coef_[0])
55
56
57
58 next_index = monthly_avg['MonthIndex'].max() + 1
59 prediction = model.predict([[next_index]])
60
61 print("\nPredicted Average Steps Next Month:", int(prediction[0]))
62
63
64 plt.figure()
65 plt.scatter(monthly_avg['MonthLabel'], y)
66 plt.plot(monthly_avg['MonthLabel'], model.predict(X))
67
68 plt.xlabel("Month")
69 plt.ylabel("Average Steps")
70 plt.title("Monthly Average Steps Prediction")
71 plt.xticks(rotation=45)
72 plt.tight_layout()
73 plt.show()

```

Output:



Predicted Average Steps Next Month: 7214