Name: Shivpratik Hande Class:D15C/14

Experiment 5

Aim:
Perform Regression Analysis using Scipy and Scikit-learn.

Problem Statement:

1. Perform Logistic Regression to determine the relationship between variables.
2. Apply a Regression Model technique to predict the data based on the given dataset.

Dataset Description:

The dataset contains more than 1 lakh instances, fulfilling the requirement for Big Data analysis. The following are the key features used in the regression models:

- **Cycle_Index**: Index of the cycle during battery charging/discharging.
- **Discharge Time (s)**: Time taken for battery discharge.
- **Decrement 3.6-3.4V (s)**: Time decrements between specific voltage levels.
- **Max. Voltage Discharge (V)**: Maximum voltage during discharge.
- **Min. Voltage Charge (V)**: Minimum voltage during charge.
- **Time at 4.15V (s)**: Time spent at a specific voltage level.
- **Time constant current (s)**: Duration of constant current phase.
- **Charging time (s)**: Total time taken for charging.
- **RUL (Remaining Useful Life)**: Predicted remaining life of the battery.

The dataset is preprocessed to remove missing values and scale numerical features before regression analysis.

Theory & Mathematical Background

1. Linear Regression:

Linear Regression is a fundamental supervised learning algorithm that models the relationship between a dependent variable (y) and one or more independent variables (X) by fitting a linear equation. It is widely used for predictive modeling where the goal is to establish a linear relationship between input and output variables.

Mathematical Representation:
The equation for multiple linear regression is:
y=β0+β1x1+β2x2+...+βnxny

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

Where:

- y = Dependent variable (Discharge Time (s))
- $X_1, X_2, \ldots, X_n$ = Independent variables
- β_0 = Intercept (constant term)
- $\beta_1, \beta_2, \ldots, \beta_n$ = Coefficients (weights assigned to independent variables)
- ϵ = Error term (random noise or variability unexplained by the model)

The coefficients (β) are estimated using the **Ordinary Least Squares (OLS) method**, which minimizes the sum of squared residuals:

$$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Name: Shivpratik Hande Class:D15C/14

2. Logistic Regression:

Logistic Regression is used when the dependent variable is binary (classification problem). Instead of predicting a continuous value, it predicts the probability that a given input belongs to a specific category (0 or 1).

Mathematical Representation:
Instead of a linear function, logistic regression uses the sigmoid function to map the predictions between 0 and 1:

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+...+\beta_n X_n)}}$$

Where:

- $P(Y=1|X)$ is the probability of the output being class **1**
- $\beta_0, \beta_1, ..., \beta_n$ are the model parameters
- e is the mathematical constant (~2.718)

Taking the **logit transformation**, we obtain the following linear equation:

$$log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n$$

The model is trained using **Maximum Likelihood Estimation (MLE)**, which finds the parameters that maximize the likelihood of the observed data.

Name: Shivpratik Hande Class:D15C/14

Comparison of Linear & Logistic Regression:

Feature	Linear Regression	Logistic Regression
Output Type	Continuous (real numbers)	Probability (0 to 1)
Used for	Regression (Prediction of values)	Classification (Binary categories)
Model Type	Linear	Non-linear (Sigmoid)
Loss Function	Mean Squared Error (MSE)	Log Loss (Cross-Entropy)
Optimization	Ordinary Least Squares (OLS)	Maximum Likelihood Estimation (MLE)

Output:

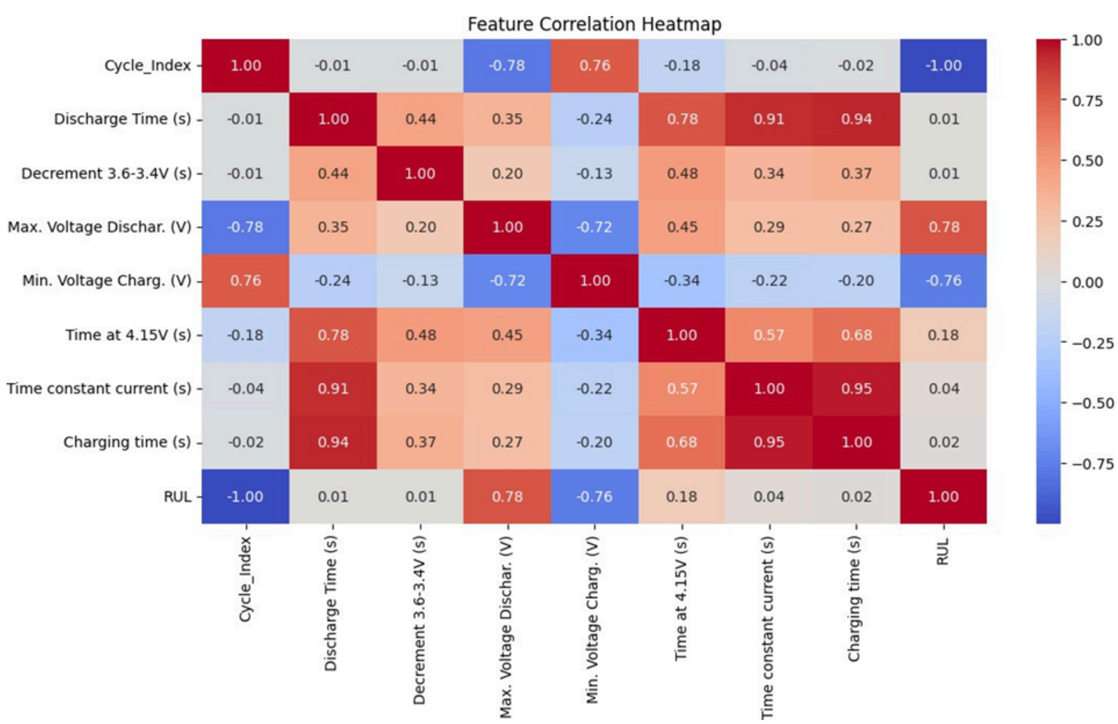
1. **Correlation Heatmap**:
   The heatmap shows correlations between features, highlighting strong positive relationships (e.g., Charging Time & Discharge Time: 0.94) and negative correlations (e.g., Cycle Index & RUL: -1.00). It confirms that battery aging affects discharge characteristics and RUL.

```
print("Missing Values:\n", df.isnull().sum())

df = df.dropna()

plt.figure(figsize=(12, 6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Feature Correlation Heatmap")
plt.show()
```

```
Missing Values:
 Cycle_Index                  0
Discharge Time (s)           0
Decrement 3.6-3.4V (s)       0
Max. Voltage Dischar. (V)    0
Min. Voltage Charg. (V)      0
Time at 4.15V (s)            0
Time constant current (s)    0
Charging time (s)            0
RUL                          0
dtype: int64
```



2. **Binary Classification using Logistic Regression**:

   ○ A new target variable `RUL_Class` is created by comparing RUL to the median, turning it into a classification task.

   ○ Selected features include: Cycle_Index, voltage measurements, and charging times.

   ○ The dataset is split into 80% training and 20% testing using `train_test_split` with `random_state=42`.

   ○ A logistic regression model is trained, but a **ConvergenceWarning** is triggered, suggesting:

      ■ Increase `max_iter`

      ■ Use data standardization

      ■ Try alternative solvers like `saga` or `lbfgs`

```
df['RUL_Class'] = (df['RUL'] > df['RUL'].median()).astype(int)
```
✓ 0.0s Python

```
X = df[['Cycle_Index', 'Decrement 3.6-3.4V (s)', 'Max. Voltage Dischar. (V)',
        'Min. Voltage Charg. (V)', 'Time at 4.15V (s)', 'Time constant current (s)', 'Charging time (s)']]
y = df['RUL_Class']
```
✓ 0.0s Python

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```
✓ 0.0s Python

```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
```
✓ 0.5s Python

```
C:\Users\adity\AppData\Roaming\Python\Python312\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: l
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Name: Shivpratik Hande Class:D15C/14

3. **Model Evaluation (Logistic Regression):**

   - Predictions are made using the test set (`y_pred`).
   - **Accuracy** is ~99.5% (very high).
   - A **confusion matrix** shows correct classifications and minimal misclassifications.
   - **Classification report** shows high precision, recall, and F1-scores.
   - However, the very low error rate may suggest overfitting due to class imbalance or high separability.

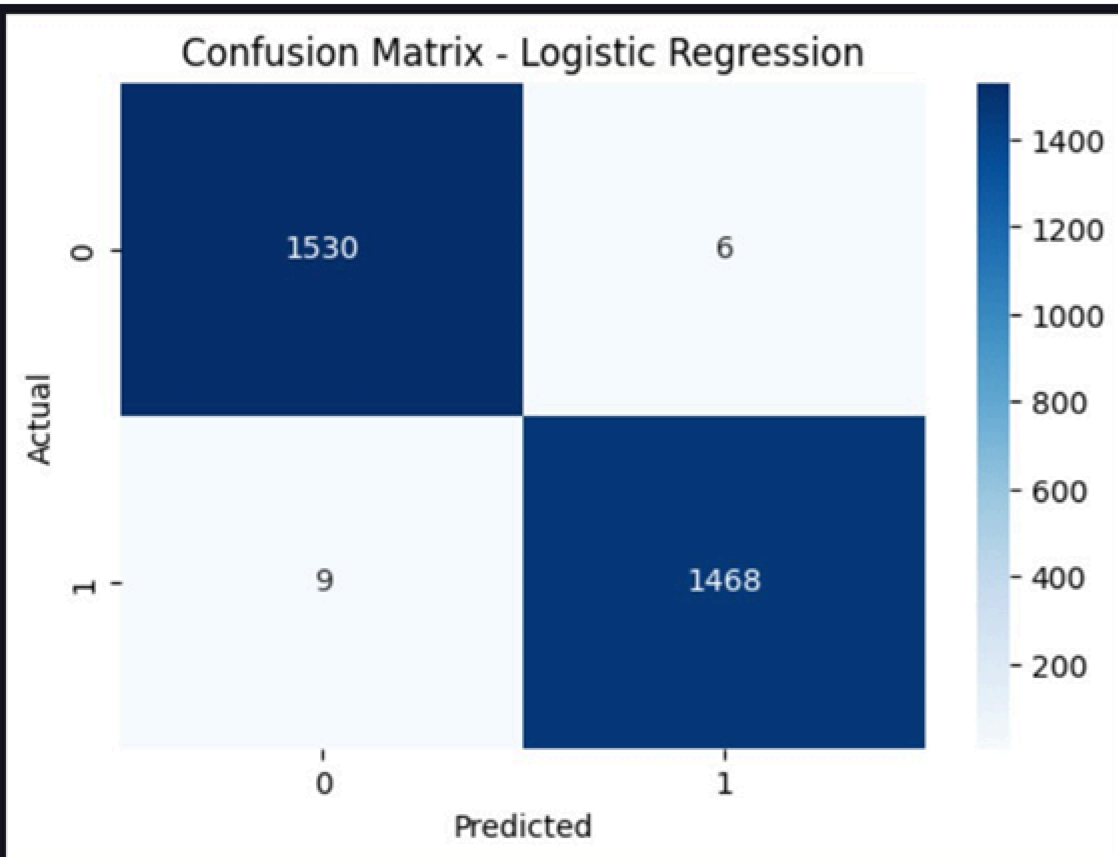
```
y_pred = log_reg.predict(X_test)
```
✓ 0.0s

```
accuracy = accuracy_score(y_test, y_pred)
print("Logistic Regression Accuracy:", accuracy)
```
✓ 0.0s

```
Logistic Regression Accuracy: 0.9950215731828742
```

```
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Logistic Regression")
plt.show()

print("Classification Report:\n", classification_report(y_test, y_pred))
```
✓ 0.2s




```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1536
           1       1.00      0.99      0.99      1477

    accuracy                           1.00      3013
   macro avg       1.00      1.00      1.00      3013
weighted avg       1.00      1.00      1.00      3013
```

---

4. **Linear Regression for Discharge Time Prediction**:
   - Model is trained to predict discharge time.
   - Dataset is split into train/test sets.

- Model performance evaluated using **MSE** and **R² score**.

```python
X = df[['Cycle_Index', 'Decrement 3.6-3.4V (s)', 'Max. Voltage Dischar. (V)',
        'Min. Voltage Charg. (V)', 'Time at 4.15V (s)', 'Time constant current (s)', 'Charging time (s)']]
y = df['Discharge Time (s)']
```
✓ 0.0s

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```
✓ 0.0s

```python
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
```
✓ 0.0s

```
LinearRegression
LinearRegression()
```

```python
y_pred = linear_reg.predict(X_test)
```
✓ 0.0s

---

5. **Model Performance (Linear Regression):**
   - **R^2 score** ≈ 0.93: Model explains 93% variance in discharge time.
   - **MSE** is large → potential outliers or large individual prediction errors.
   - **Scatter plot** shows upward trend in predictions but errors for high discharge times suggest:
     - Outliers or heteroscedasticity
     - Linear model limitations
     - Consider: feature scaling, polynomial regression, or models like Random Forest or Gradient Boosting.

Name: Shivpratik Hande Class:D15C/14

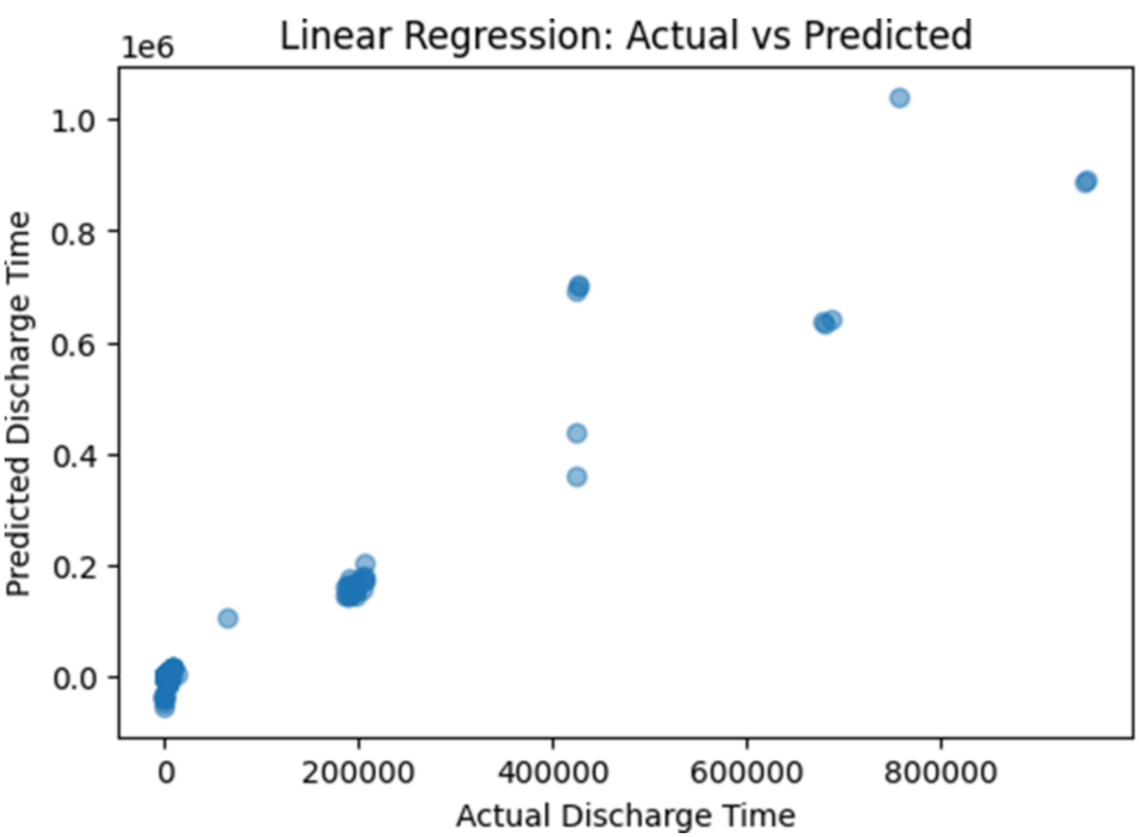
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Linear Regression MSE:", mse)
print("Linear Regression R² Score:", r2)
```

✓ 0.0s

```
Linear Regression MSE: 127024271.8764475
Linear Regression R² Score: 0.9315065475881482
```

```
plt.figure(figsize=(6, 4))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel("Actual Discharge Time")
plt.ylabel("Predicted Discharge Time")
plt.title("Linear Regression: Actual vs Predicted")
plt.show()
```

✓ 0.2s



Conclusion:

- **Linear Regression** is suitable for predicting continuous values like discharge time.
    - High R^2 (~0.93) but outliers/non-linearity affect accuracy.
- **Logistic Regression** is used for classification and performed well but may overfit.
    - Not ideal for continuous outputs.
- Model performance can be enhanced by:
    - Handling outliers
    - Feature engineering
    - Advanced models (Random Forest, ensemble methods)

Understanding the strengths and limitations of each technique aids in better machine learning predictions and decision-making.