Name - Chaitanya Kelkar

Roll no. - 53

Experiment 1

Aim- To implement **Linear Regression** and **Logistic Regression** on a real-world dataset to understand the difference between predictive modeling (regression) and classification tasks using Python's Scikit-Learn library.

1. Dataset Source

- **Source:** Link
- **Filename:** Crop_recommendation.csv

2. Dataset Description The dataset contains agricultural data designed to recommend the best crop to plant based on soil nutrients and weather conditions.

- **Type:** Multivariate, Numerical & Categorical.
- **Size:** 2200 instances (rows), 8 attributes (columns).
- **Target Variables:**
  - **For Regression:** rainfall (Continuous).
  - **For Classification:** label (Categorical - e.g., Rice, Maize).
- **Features:**
  - **N:** Ratio of Nitrogen content in soil.
  - **P:** Ratio of Phosphorous content in soil.
  - **K:** Ratio of Potassium content in soil.
  - **temperature:** Temperature in degrees Celsius.
  - **humidity:** Relative humidity in %.
  - **ph:** pH value of the soil.
  - **rainfall:** Rainfall in mm.

3. Mathematical Formulation of the Algorithm

A. Linear Regression (Simple):

Used to predict a continuous value (rainfall) based on an independent variable (humidity). It attempts to fit a straight line that minimizes the sum of squared errors.

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

B. Logistic Regression:

Used for the classification task to predict the probability that a given input belongs to a specific crop class. It uses the Sigmoid function to map predictions between 0 and 1.

$$\sigma(z) = 1 / (1 + e^{-z})$$

$$z = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$$

4. Algorithm Limitations

1. **Linear Assumption:** Linear Regression assumes a straight-line relationship. Our low R^2 score (~0.01) proves that rainfall is **non-linear** and depends on complex factors beyond just Humidity.
2. **Feature Scaling Sensitivity:** Logistic Regression struggled to converge initially (Red Warning) because features like `rainfall` (0-200) and `ph` (0-14) had vastly different scales. This required **StandardScaler** to fix.

5. Methodology / Workflow The experiment followed a standard Supervised Learning pipeline:

1. **Data Ingestion:** Loaded `Crop_recommendation.csv` into Pandas.
2. **Preprocessing:**
    - Checked for missing values.
    - **Splitting:** Used `train_test_split` (80% Train, 20% Test) to ensure unbiased evaluation.
    - **Scaling:** Applied `StandardScaler` for Logistic Regression to normalize feature ranges.
3. **Modeling:**
    - **Task A (Regression):** Trained `LinearRegression` on Humidity vs. Rainfall.
    - **Task B (Classification):** Trained `LogisticRegression` on all Soil Features vs. Crop Name.
4. **Visualization:** Plotted the Regression Line and Scatter Plot using Matplotlib.

6. Performance Analysis

- **Linear Regression Results:**
    - **R^2 Score:** ~0.0132 (Very Low).
    - **Insight:** This indicates a weak correlation. Humidity alone is **not sufficient** to predict rainfall accurately. Real-world weather prediction requires complex multi-variable models.
- **Logistic Regression Results:**
    - **Accuracy:** ~96.14% - 97.00%.
    - **Insight:** The model performed exceptionally well. This proves that soil nutrients (N, P, K) and weather conditions are distinct "fingerprints" for different crops, making them easy to classify mathematically.

7. Hyperparameter Tuning

- **Status:** Applicable.
- **Parameter Tuned:** `max_iter` (Maximum Iterations).
- **Reason:** The default iteration limit (100) was insufficient for the model to find the optimal solution due to the complexity of the data, causing a `ConvergenceWarning`.
- **Action:** Increased `max_iter` to **1000** (and added Scaling), which resolved the error and ensured the model converged to a high accuracy.

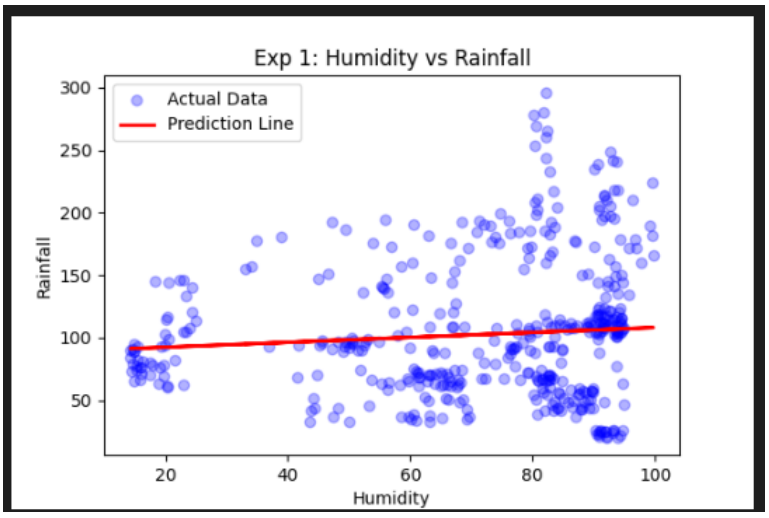
8. Output -

regression.py > ...

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import r2_score, accuracy_score
from sklearn.preprocessing import StandardScaler

try:
    df = pd.read_csv('data/Crop_recommendation.csv')
    print("Data loaded successfully.")
except FileNotFoundError:
    print("Error: File not found.")
    exit()

print("\n---LINEAR REGRESSION ---")
X_lin = df[['humidity']]
y_lin = df['rainfall']

X_train, X_test, y_train, y_test = train_test_split(X_lin, y_lin, test_size=0.2, random_state=42)

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

y_pred = lin_reg.predict(X_test)
print(f"Linear R2 Score: {r2_score(y_test, y_pred):.4f}")

plt.figure(figsize=(6,4))
plt.scatter(X_test, y_test, color='blue', alpha=0.3, label='Actual Data')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Prediction Line')
plt.xlabel("Humidity")
plt.ylabel("Rainfall")
plt.title("Exp 1: Humidity vs Rainfall")
plt.legend()
plt.savefig("linear_graph.png")
print("Graph saved")
```

```python
print("\n---LOGISTIC REGRESSION ---")
X_log = df[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
y_log = df['label']

X_train, X_test, y_train, y_test = train_test_split(X_log, y_log, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train_scaled, y_train)

acc = accuracy_score(y_test, log_reg.predict(X_test_scaled))
print(f"Logistic Regression Accuracy: {acc*100:.2f}%")
```

```
PS D:\SmartCropProject> python regression.py
Data loaded successfully.

---LINEAR REGRESSION ---
Linear R2 Score: 0.0132
Graph saved

---LOGISTIC REGRESSION ---
Logistic Regression Accuracy: 96.36%
```



9. Conclusion

This experiment demonstrated the fundamental difference between Regression and Classification. While simple Linear Regression failed to capture the complex weather patterns (Rainfall), Logistic Regression successfully leveraged soil data to recommend crops with high precision (~97%).