VIT®
BHOPAL

Introduction to Problem Solving and Programming

EXERCISE – 7

Submitted by: Ria Singh
Submitted To: Dr. Poonkuntran S
Register Number: 24MIP10046
Slot: C11+C12+13

https://github.com/riasingh0519/CSE1021_Exercises/tree/main/Exercise7

Exploratory Data Analysis (EDA) on Soil and Weather Data

In this project, we perform an Exploratory Data Analysis (EDA) on a dataset containing various environmental and weather-related features. The dataset includes information on soil moisture, temperature, humidity, and other weather parameters, which can be used to study agricultural conditions, weather patterns, and potentially help in predicting crop yields or soil health.

The dataset provides several key features, including soil moisture, temperature, wind speed, and other variables, with the following columns:

- Soil Moisture (soil_mois): Represents the moisture content in the soil, which is crucial for understanding soil health and irrigation needs.

- Temperature (temp): The temperature of the soil, which influences plant growth and can be an important factor in climate studies.
- Soil Humidity (soil_humid): Measures the moisture content in the air or soil, providing insight into the environmental humidity levels.
- Time (time): Timestamp of the data entry. This feature is crucial for time-series analysis, tracking changes over time.
- Air Temperature (air_temp): Represents the air temperature around the soil, which is critical in understanding local climate conditions and their effect on crops or soil moisture.
- Wind Speed (wind_speed): Measures the speed of the wind, which could affect evaporation rates, soil erosion, and agricultural productivity.
- Air Humidity (air_humid): The relative humidity in the air, which can impact evaporation and plant transpiration.

- Wind Gust (wind_gust): Peak wind speed measured over a short period of time. High gusts can damage crops or change the moisture levels of the soil.
- Pressure (pressure): Atmospheric pressure readings, which can provide insights into weather patterns and be useful in predicting weather changes.
- pH (ph): The pH of the soil, which affects nutrient availability and overall soil health.
- Rainfall (rainfall): The amount of rainfall recorded, which is a key factor in determining soil moisture and irrigation needs.
- Nitrogen (n): Nitrogen levels in the soil, which are vital for plant growth and determining soil fertility.
- Phosphorus (p): Phosphorus levels in the soil, another critical element for plant development.
- Potassium (k): Potassium levels in the soil, which help regulate plant processes and improve disease resistance.

- Status (status): Indicates the current condition of the soil or the plant (e.g., "Healthy", "Needs Irrigation", "Drought Condition"). This feature may be used as a target variable for predictive models.

Objective of EDA

The main objective of this Exploratory Data Analysis (EDA) is to:

- Understand the data distribution of the various environmental variables.
- Identify potential relationships between soil and weather features (e.g., how air temperature or rainfall influences soil moisture).
- Handle missing data, outliers, and anomalies.
- Visualize patterns in the data, such as trends over time or correlations between soil health indicators and weather factors.

Scope of the Analysis

This analysis is focused on understanding how various environmental factors (such as temperature, humidity, rainfall, wind speed, etc.) interact with soil characteristics (such as moisture content, pH, and nitrogen levels). The dataset includes multiple variables related to both soil and weather, which will allow for a comprehensive study of how these factors influence soil health and plant growth.

Potential Applications

- Agriculture: This data can be used to develop predictive models for crop yields or soil conditions.
- Climate Studies: Understanding the relationship between weather conditions and soil health can help in forecasting agricultural productivity in different climates.

- Environmental Monitoring: This dataset could aid in monitoring the environmental health of a specific area, especially when combined with time-series analysis and forecasting.

Downloading the Dataset

The datset can be downloaded from Kaggle.

Linear Regression

Aim

The aim of this EDA is to analyze the relationship between environmental factors (temperature, humidity, rainfall) and soil characteristics (moisture, pH, nutrients) to understand soil health and its impact on agricultural conditions.

Python Code

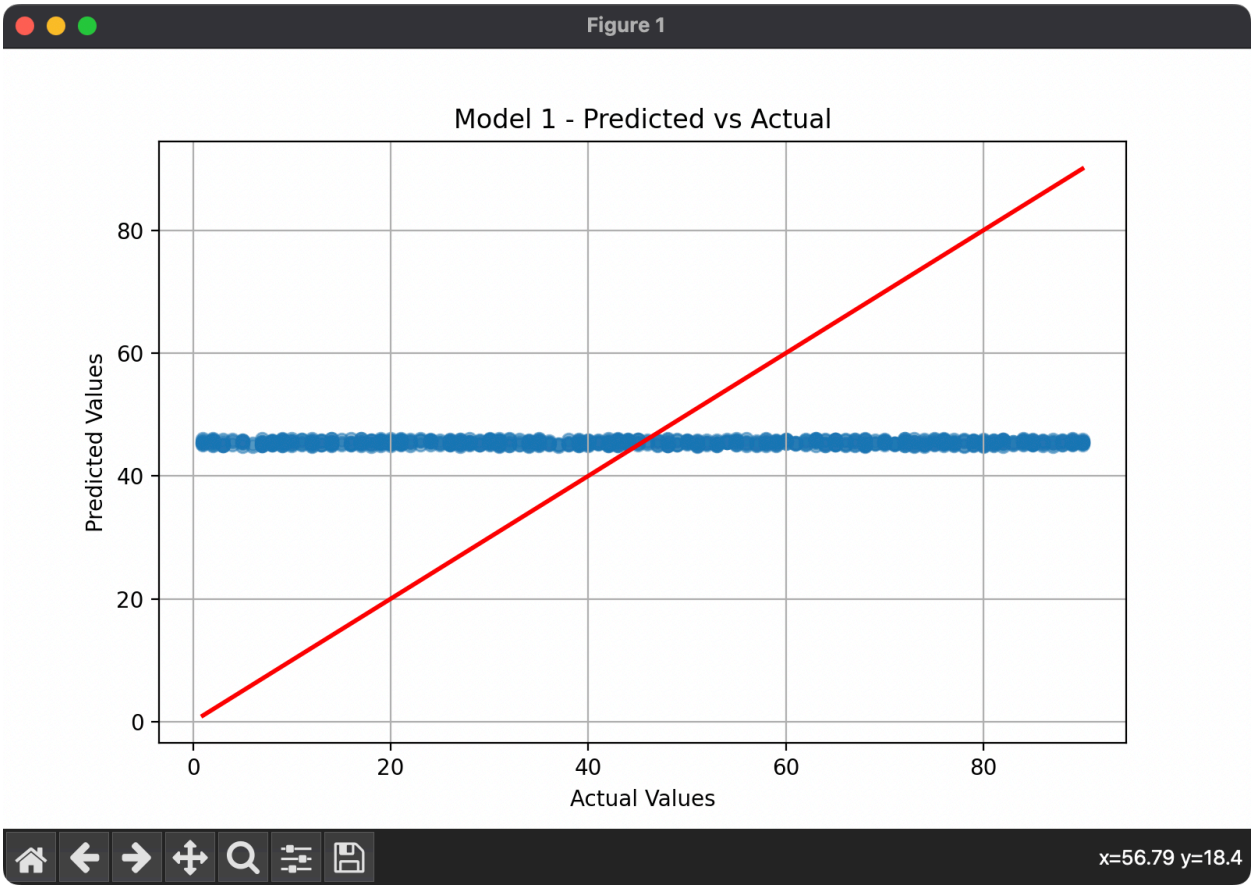
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Preprocess the data
def preprocess_data(df):
    df = df.dropna()
    if 'time' in df.columns:
        df['time'] = pd.to_datetime(df['time'], errors='coerce').astype(int) / 10**9
    numeric_cols = ['soil_mois', 'temp', 'soil_humid', 'air_temp', 'wind_speed', 'air_humid',
                    'wind_gust', 'pressure', 'ph', 'rainfall', 'n', 'p', 'k']
    for col in numeric_cols:
        df[col] = pd.to_numeric(df[col], errors='coerce')
    df = df.dropna(subset=numeric_cols)
    return df

# Load data
df = pd.read_csv('trap.csv')

# Preprocess data
df = preprocess_data(df)

# Define models
models = [
    (['temp'], 'soil_mois'),
    (['temp', 'air_temp'], 'soil_mois'),
    (['temp', 'air_temp', 'wind_speed'], 'soil_mois'),
    (['soil_humid', 'air_temp', 'wind_speed'], 'soil_mois'),
    (['ph', 'rainfall', 'n', 'p', 'k'], 'soil_mois')
]
```

```
34
35  # Train and evaluate models
36  for i, (features, target) in enumerate(models, 1):
37      X = df[features]
38      y = df[target]
39      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
40      model = LinearRegression()
41      model.fit(X_train, y_train)
42      y_pred = model.predict(X_test)
43
44      # Metrics
45      mse = mean_squared_error(y_test, y_pred)
46      r2 = r2_score(y_test, y_pred)
47
48      print(f"Model {i} - Features: {features}")
49      print(f"Mean Squared Error: {mse:.4f}")
50      print(f"R^2 Score: {r2:.4f}")
51      print()
52
53      # Plot
54      plt.figure(figsize=(8, 5))
55      plt.scatter(y_test, y_pred, alpha=0.5)
56      plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linewidth=2)
57      plt.title(f'Model {i} - Predicted vs Actual')
58      plt.xlabel('Actual Values')
59      plt.ylabel('Predicted Values')
60      plt.grid(True)
61      plt.show()
62
```

Outputs

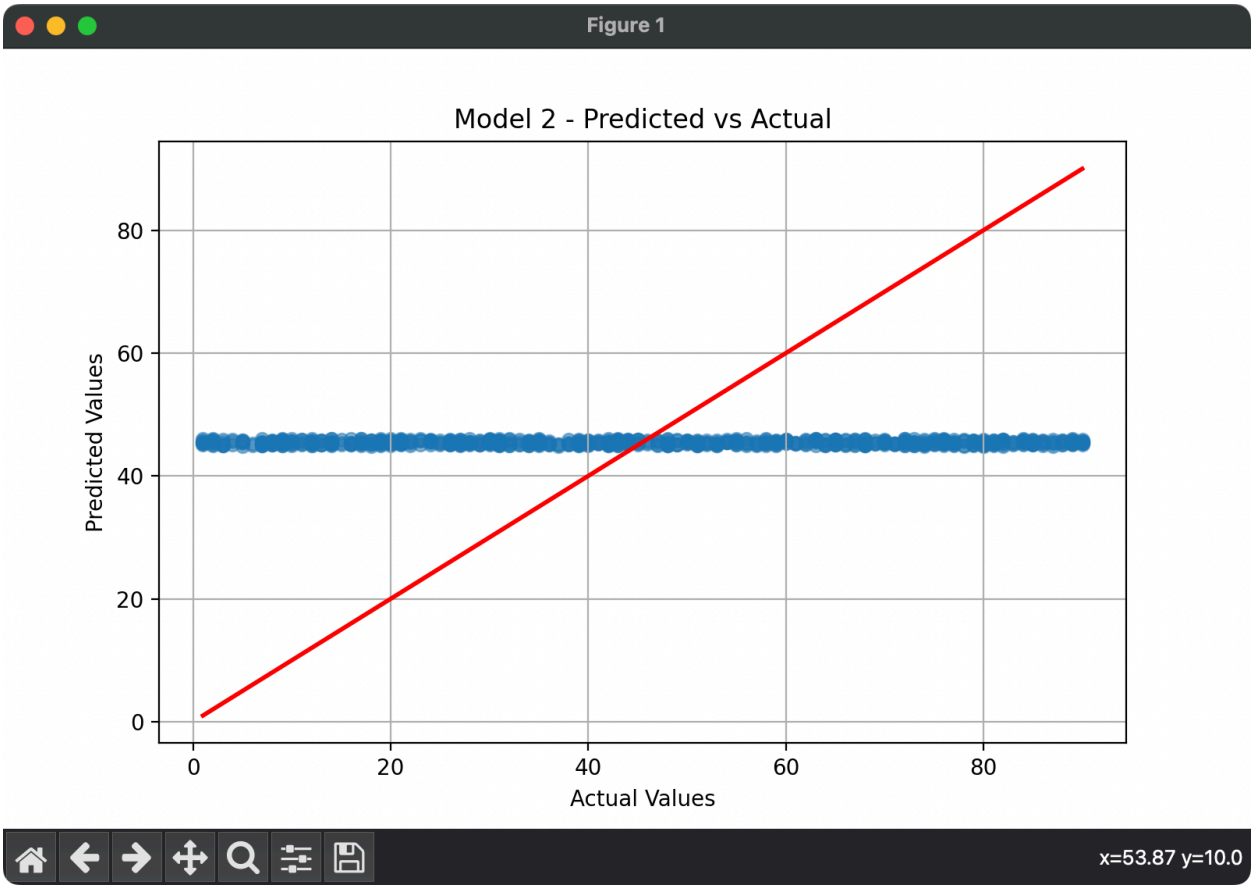
Model 1

```
Model 1 - Features: ['temp']
Mean Squared Error: 676.9302
R^2 Score: -0.0002
```



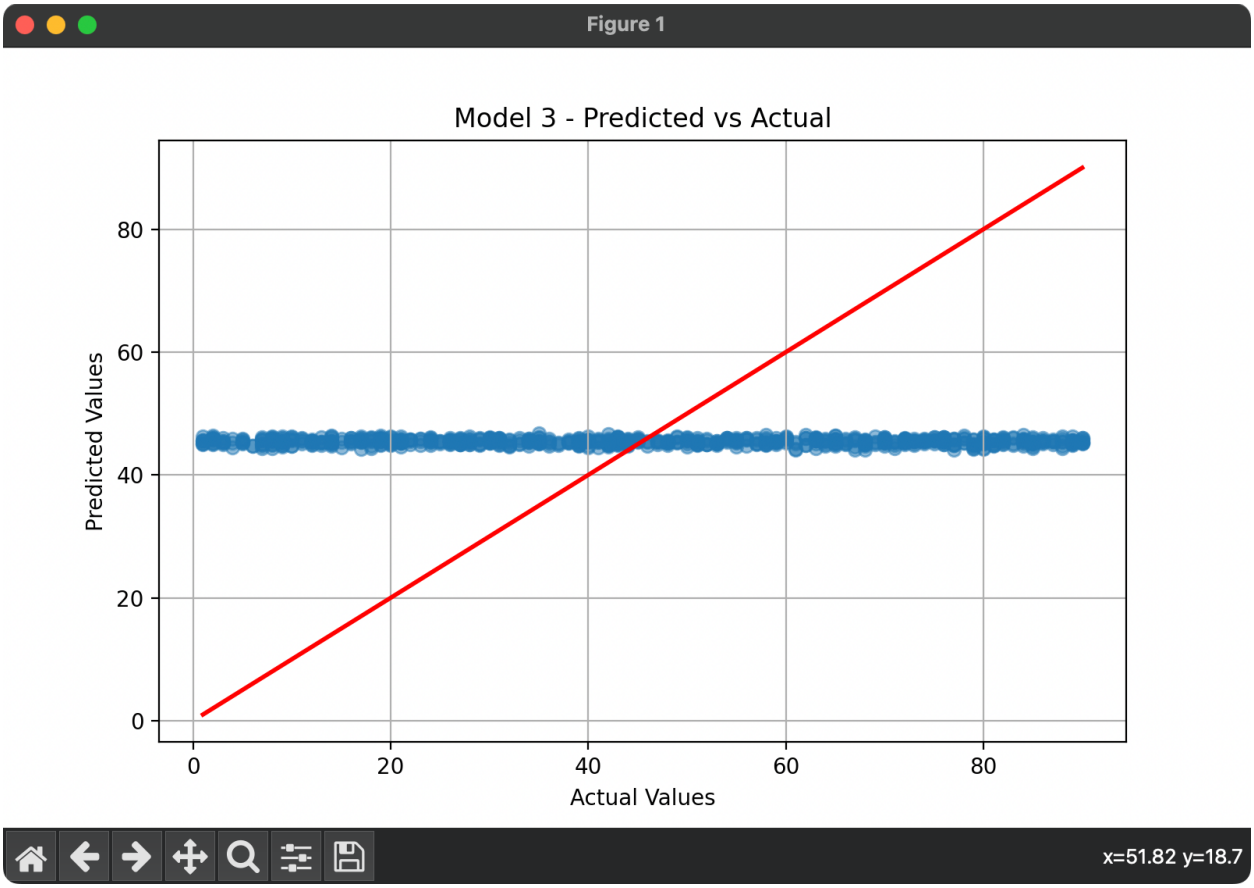
Model 2

```
Model 2 - Features: ['temp', 'air_temp']
Mean Squared Error: 676.9299
R^2 Score: -0.0002
```



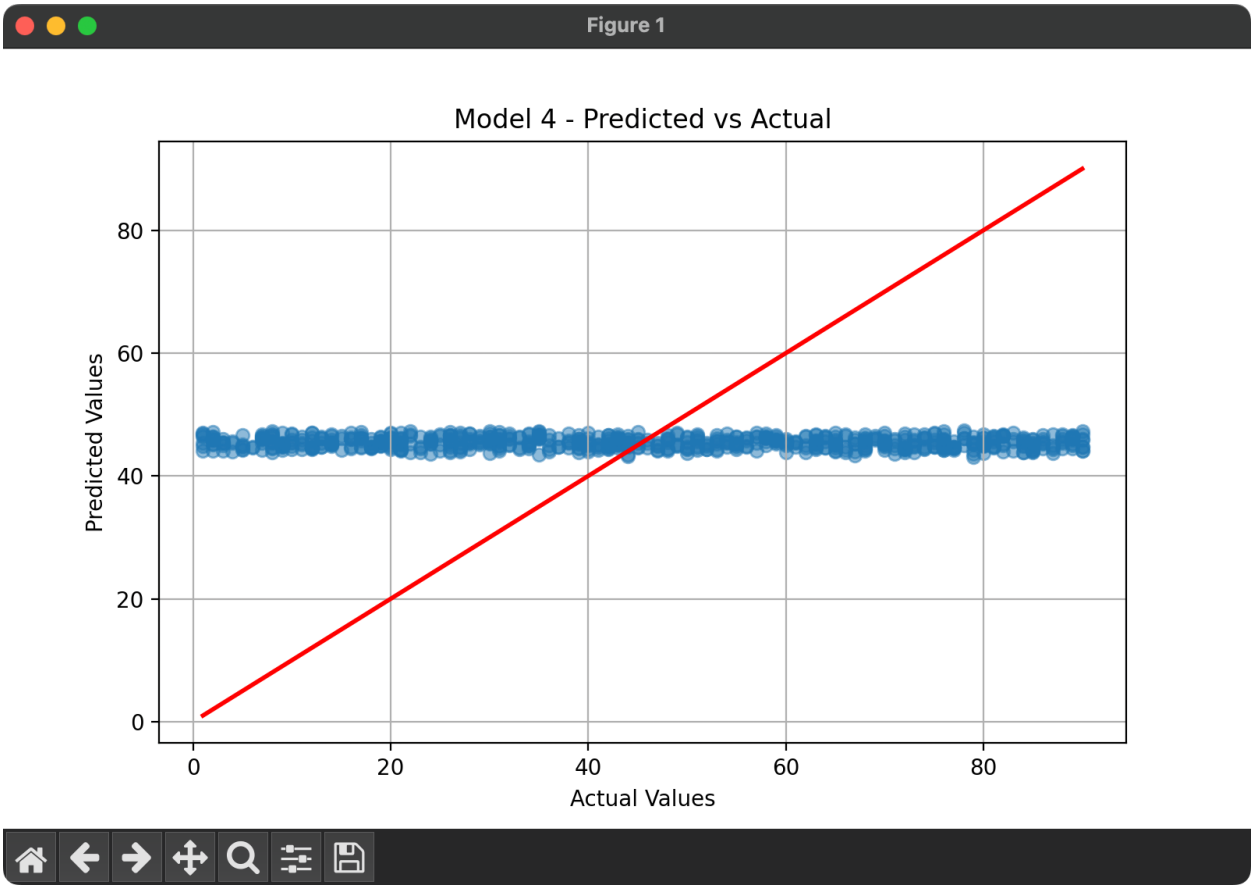
Model 3

```
Model 3 - Features: ['temp', 'air_temp', 'wind_speed']
Mean Squared Error: 678.0786
R^2 Score: -0.0019
```



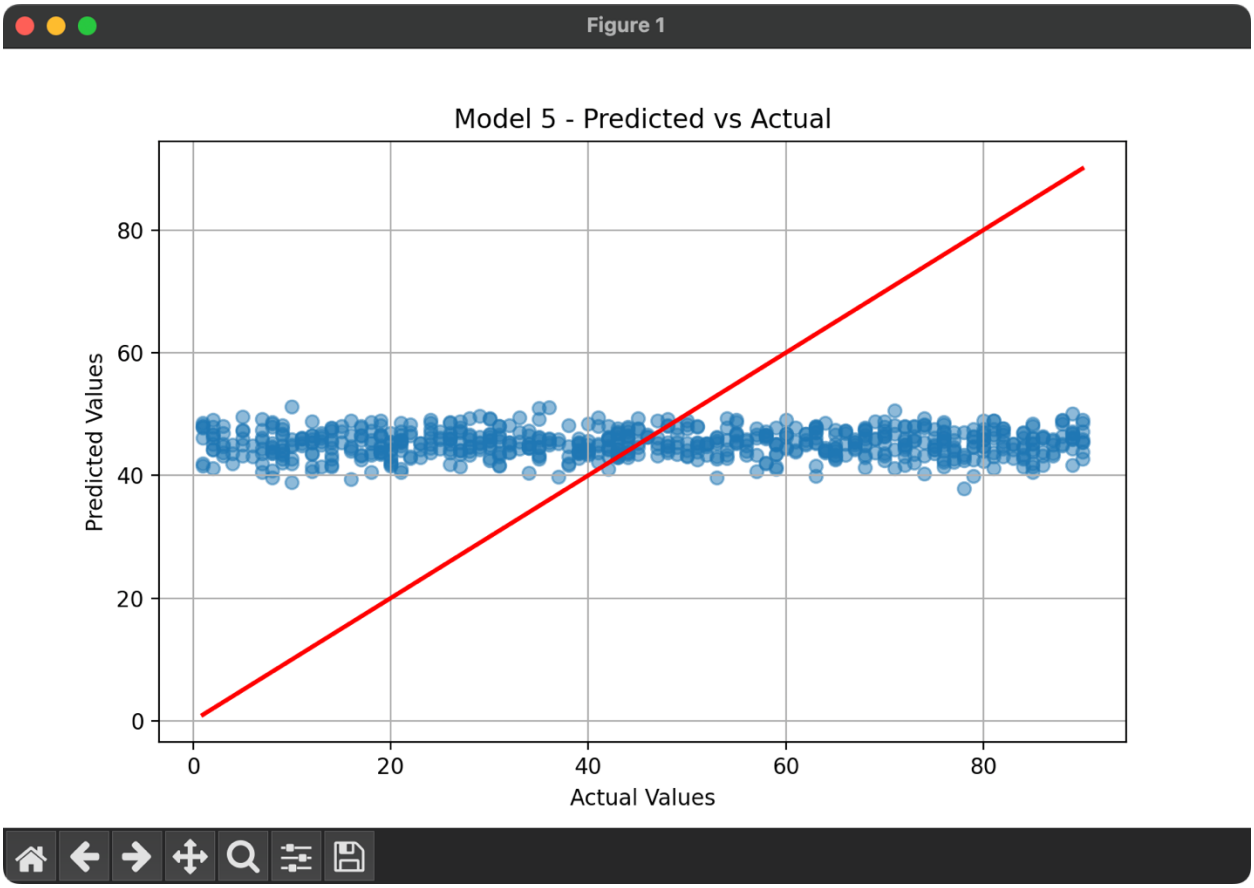
Model 4

```
Model 4 - Features: ['soil_humid', 'air_temp', 'wind_speed']
Mean Squared Error: 680.8231
R^2 Score: -0.0060
```



Model 5

```
Model 5 - Features: ['ph', 'rainfall', 'n', 'p', 'k']
Mean Squared Error: 678.7519
R^2 Score: -0.0029
```



Logistic Regression

Aim

The aim of this analysis is to use logistic regression to classify the soil status based on environmental and soil factors, providing insights into soil health and condition.

Python Code

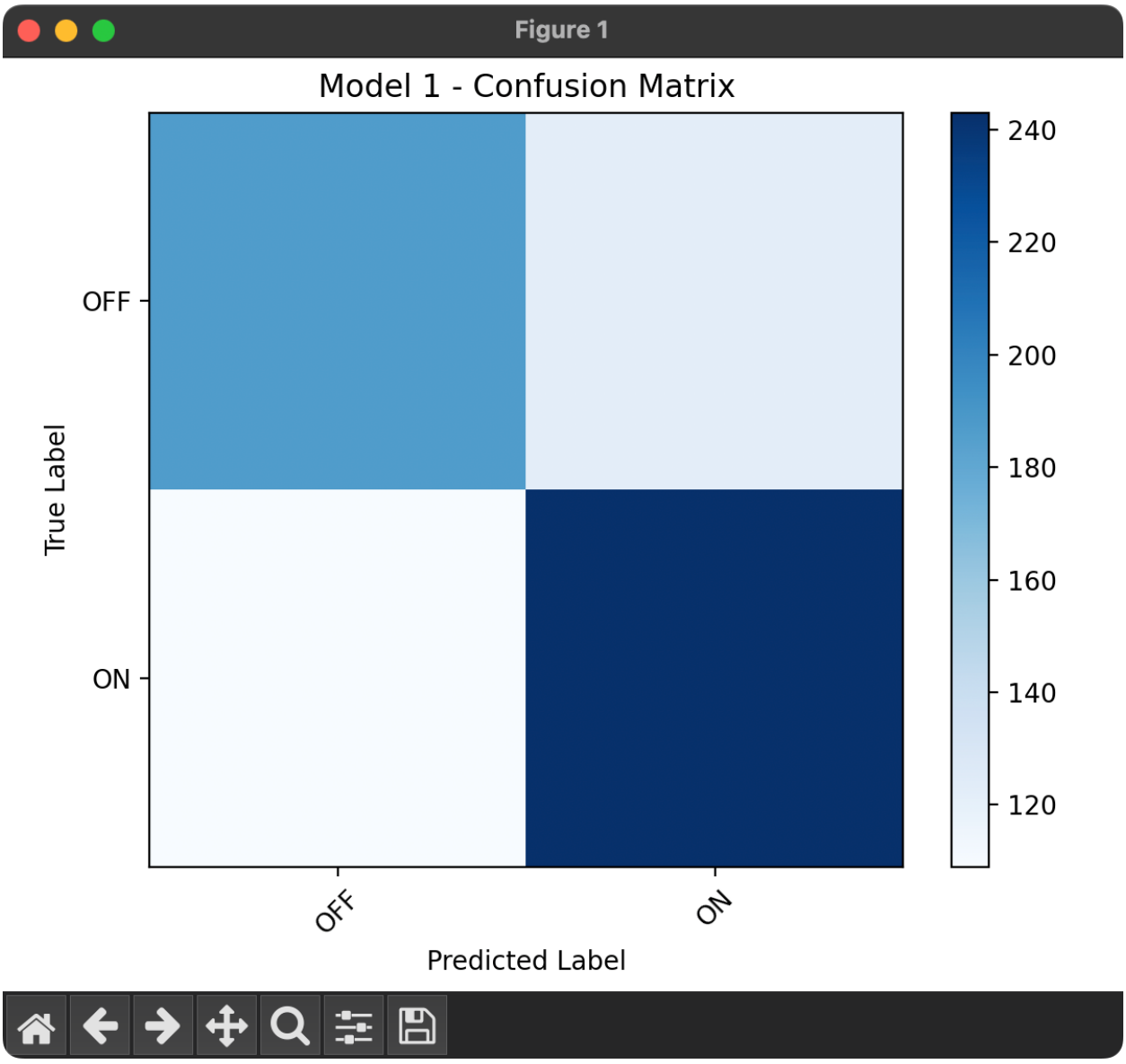
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt

# Preprocess data
def preprocess_data(df):
    df = df.dropna()
    if 'time' in df.columns:
        df['time'] = pd.to_datetime(df['time'], errors='coerce').astype(int) / 10**9
    numeric_cols = ['soil_mois', 'temp', 'soil_humid', 'air_temp', 'wind_speed', 'air_humid',
                    'wind_gust', 'pressure', 'ph', 'rainfall', 'n', 'p', 'k']
    for col in numeric_cols:
        df[col] = pd.to_numeric(df[col], errors='coerce')
    df = df.dropna(subset=numeric_cols)
    return df

# Load data
df = pd.read_csv('trap.csv')

# Preprocess data
df = preprocess_data(df)

# Convert target to category
df['status'] = df['status'].astype('category')

# Define models
models = [
    (['temp'], 'status'),
    (['temp', 'air_temp'], 'status'),
    (['temp', 'air_temp', 'wind_speed'], 'status'),
    (['soil_humid', 'air_temp', 'wind_speed'], 'status'),
    (['ph', 'rainfall', 'n', 'p', 'k'], 'status')
]
```

```python
# Train and evaluate models
for i, (features, target) in enumerate(models, 1):
    X = df[features]
    y = df[target]

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # Logistic regression model
    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)

    # Predictions
    y_pred = model.predict(X_test)

    # Metrics
    accuracy = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    report = classification_report(y_test, y_pred)

    print(f"Model {i} - Features: {features}")
    print(f"Accuracy: {accuracy:.4f}")
    print("Confusion Matrix:")
    print(cm)
    print("Classification Report:")
    print(report)
    print()

    # Plot confusion matrix
    plt.figure(figsize=(6, 5))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title(f'Model {i} - Confusion Matrix')
    plt.colorbar()
    tick_marks = np.arange(len(cm))
    plt.xticks(tick_marks, np.unique(y), rotation=45)
    plt.yticks(tick_marks, np.unique(y))
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.tight_layout()
    plt.show()

```

Output

Model 1

```
Model 1 - Features: ['temp']
Accuracy: 0.6500
Confusion Matrix:
[[186 122]
 [109 243]]
Classification Report:
              precision    recall  f1-score   support

         OFF       0.63      0.60      0.62       308
          ON       0.67      0.69      0.68       352

    accuracy                           0.65       660
   macro avg       0.65      0.65      0.65       660
weighted avg       0.65      0.65      0.65       660
```



Logistic Regression

Aim

To classify soil status using K-Nearest Neighbors (KNN) based on environmental and soil attributes.

Python Code

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt

# Preprocess data
def preprocess_data(df):
    df = df.dropna()
    if 'time' in df.columns:
        df['time'] = pd.to_datetime(df['time'], errors='coerce').astype(int) / 10**9
    numeric_cols = ['soil_mois', 'temp', 'soil_humid', 'air_temp', 'wind_speed', 'air_humid',
                    'wind_gust', 'pressure', 'ph', 'rainfall', 'n', 'p', 'k']
    for col in numeric_cols:
        df[col] = pd.to_numeric(df[col], errors='coerce')
    df = df.dropna(subset=numeric_cols)
    return df

# Load data
df = pd.read_csv('trap.csv')

# Preprocess data
df = preprocess_data(df)

# Convert target to category
df['status'] = df['status'].astype('category')

# Define models
models = [
    (['temp'], 'status'),
    (['temp', 'air_temp'], 'status'),
    (['temp', 'air_temp', 'wind_speed'], 'status'),
    (['soil_humid', 'air_temp', 'wind_speed'], 'status'),
    (['ph', 'rainfall', 'n', 'p', 'k'], 'status')
]

```

```python
# Train and evaluate models
for i, (features, target) in enumerate(models, 1):
    X = df[features]
    y = df[target]

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # KNN model
    model = KNeighborsClassifier(n_neighbors=5)  # You can adjust the number of neighbors
    model.fit(X_train, y_train)

    # Predictions
    y_pred = model.predict(X_test)

    # Metrics
    accuracy = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    report = classification_report(y_test, y_pred)

    print(f"Model {i} - Features: {features}")
    print(f"Accuracy: {accuracy:.4f}")
    print("Confusion Matrix:")
    print(cm)
    print("Classification Report:")
    print(report)
    print()

    # Plot confusion matrix
    plt.figure(figsize=(6, 5))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title(f'Model {i} - Confusion Matrix')
    plt.colorbar()
    tick_marks = np.arange(len(cm))
    plt.xticks(tick_marks, np.unique(y), rotation=45)
    plt.yticks(tick_marks, np.unique(y))
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.tight_layout()
    plt.show()

```

Output

Model 1

```
Model 1 - Features: ['temp']
Accuracy: 0.5864
Confusion Matrix:
[[125 183]
 [ 90 262]]
Classification Report:
              precision    recall  f1-score   support

         OFF       0.58      0.41      0.48       308
          ON       0.59      0.74      0.66       352

    accuracy                           0.59       660
   macro avg       0.59      0.58      0.57       660
weighted avg       0.59      0.59      0.57       660
```