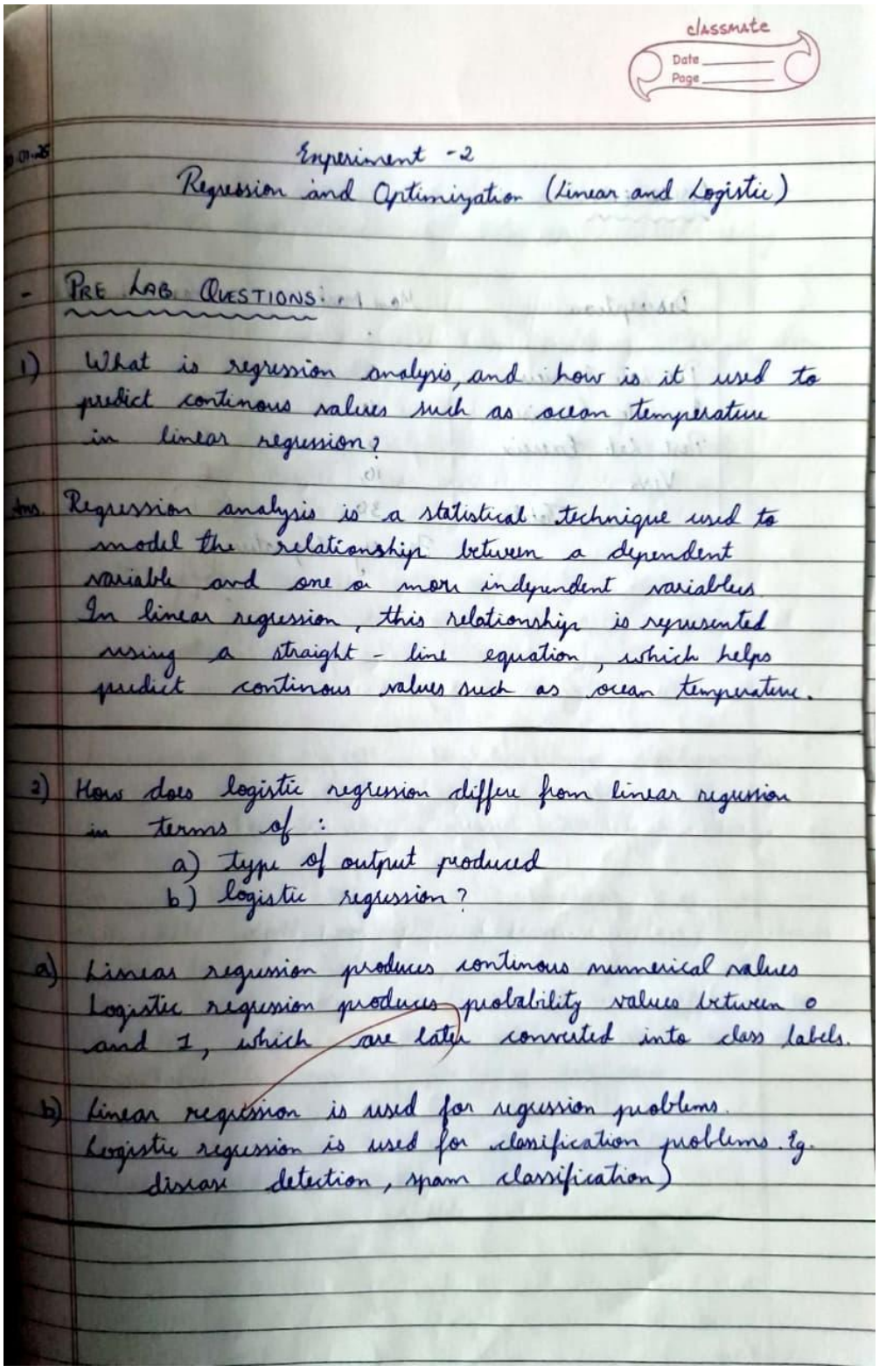
EXPT NO:2	Regression and Optimization (Linear and Logistic regression)
DATE: 30.01.2026	

PRE-LAB QUESTIONS

SUDHIKSHA 24BAD117



Experiment -2

Regression and Optimization (Linear and Logistic)

- PRE LAB QUESTIONS:

1) What is regression analysis, and how is it used to predict continous values such as ocean temperature in linear regression?

Ans. Regression analysis is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. In linear regression, this relationship is represented using a straight-line equation, which helps predict continous values such as ocean temperature.

2) How does logistic regression differ from linear regression in terms of:
   a) type of output produced
   b) logistic regression?

a) Linear regression produces continous numerical values. Logistic regression produces probability values between 0 and 1, which are later converted into class labels.

b) Linear regression is used for regression problems. Logistic regression is used for classification problems. Eg. disease detection, spam classification)

3) In a regression equation

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

what does each regression coefficient β_i indicate about the corresponding feature?

Ans. Each regression coefficient represents the change in the output variable for a one-unit change in corresponding feature, assuming all other features remain constant. The sign indicates the direction of influence, while the magnitude indicates the strength of impact

4) Why is feature scaling (standardization or normalization) necessary before applying:

a) gradient - descent based linear regression
b) logistic regression?

a) Gradient - descent - based linear regression feature scaling ensures faster and more stable convergence of gradient descent by preventing features with large values from dominating the optimization.

b) Feature scaling improves numerical stability and speeds up convergence, especially when gradient based optimization techniques are used.

5) What is a cost function, and how is it used to evaluate the performance of regression model?

Ans A cost function measures the difference between the predicted values and actual values. It provides a single numerical value that represents model error, and lower cost values indicate better model performance.

6) What is optimization in machine learning, and what does it mean to minimize a loss function?

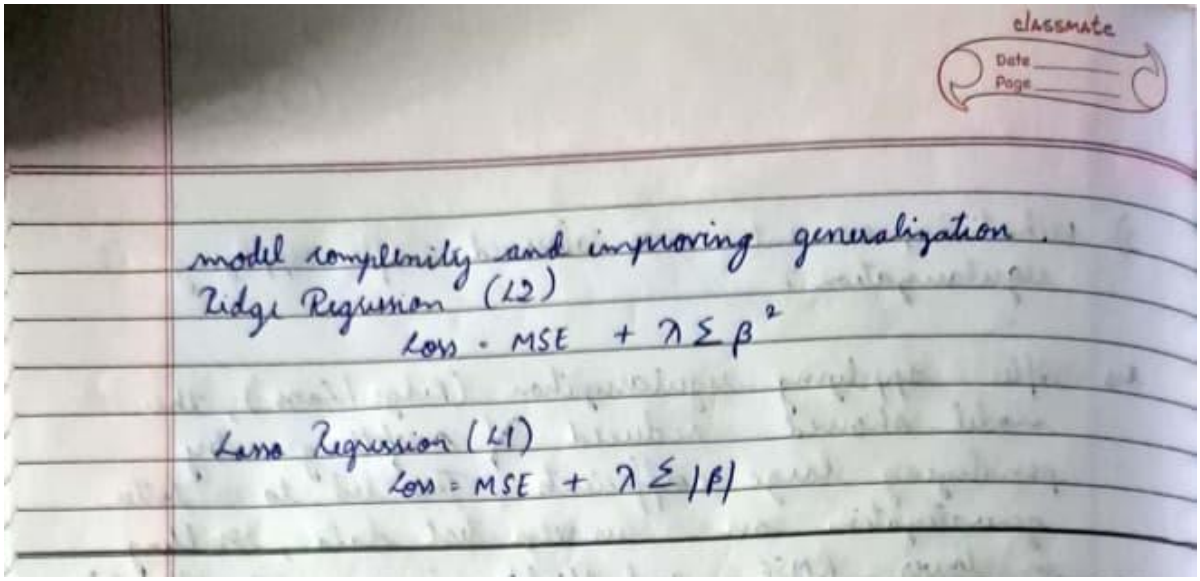
Ans. Optimization is the process of adjusting model parameters values that result in ~~smaller~~ improved performance. Minimizing a loss function means finding prediction error, leading to more accurate model.

7) Why is gradient descent prefered over solving equations directly when training regression models on large datasets?

Ans Gradient descent is prefered because solving equations directly becomes computationally expensive for large datasets. Gradient descent is more efficient, scalable, and can handle high-dimensional data effectively.

8) What is overfitting in regression models, how does regularization help reduce it?

Ans Overfitting occurs when a model learns noise in the training data and performs poorly on new data. Regularization techniques such as Ridge (L2) and Lasso (L1) add penalty terms to the loss function, reducing



IN-LAB EXERCISE
OBJECTIVE:
To implement and optimize **linear regression** and **logistic regression** models using real-world datasets and analyze their performance.

SCENARIO 1:
Predict ocean **water temperature** using environmental and depth-related features.
Dataset (Kaggle – Public):
https://www.kaggle.com/datasets/sohier/calcofi
Target Variable:

- Water Temperature (T_degC)

Sample Input Features

- Depth (m)
- Salinity
- Oxygen
- Latitude
- Longitude

IN-LAB TASKS

- Import necessary Python libraries (NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn).
- Load the CalCOFI dataset into a Pandas DataFrame
- Select relevant numerical features and target variable.
- Handle missing values using mean/median imputation.
- Perform feature scaling using StandardScaler.
- Split the dataset into training and testing sets.
- Train a **Linear Regression** model using Scikit-learn.
- Predict water temperature for test data.
- Evaluate model performance using:
  – Mean Squared Error (MSE)
  – Root Mean Squared Error (RMSE)
  – R^2 Score
- Visualize:
  – Actual vs Predicted temperature
  – Residual errors

- Optimize model performance using:
  – Feature selection
  – Regularization (Ridge / Lasso)

SCENARIO 2:
Classify whether LIC stock price will **increase (1)** or **decrease (0)** based on historical data.
Dataset (Kaggle – Public):
https://www.kaggle.com/datasets/debashis74017/lic-stock-price-data
Target Variable (Derived):
• Price Movement
– 1 → Closing price > Opening price
– 0 → Closing price ≤ Opening price
Input Features:
• Open
• High
• Low
• Volume

IN-LAB TASKS (Logistic Regression)
- Import required Python libraries.
- Load LIC stock dataset into Pandas.
- Create a binary target variable (Price Movement).
- Handle missing values.
- Perform feature scaling.
- Split the dataset into training and testing sets.
- Train a **Logistic Regression** model.
- Predict stock movement for test data.
- Evaluate classification performance using:
  – Accuracy
  – Precision
  – Recall
  – F1-Score
  – Confusion Matrix
- Visualize:
  – ROC Curve
  – Feature importance
- Optimize model using:
  – Hyperparameter tuning (C, penalty)
  – Regularization

CODE:

```python
#SCENARIO 1: Predict Ocean Water Temperature
#Sudhiksha 24BAD117

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score


df = pd.read_csv("bottle.csv", low_memory=False)
df.columns = df.columns.str.strip()
features = ['Depthm', 'Salnty', 'O2ml_L']
target = 'T_degC'

df = df[features + [target]]
df.fillna(df.mean(), inplace=True)
X = df[features]
y = df[target]


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)


lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Linear Regression Performance:")
print("MSE :", mse)
print("RMSE:", rmse)
print("R2  :", r2)

plt.figure(figsize=(6,5))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual Temperature")
plt.ylabel("Predicted Temperature")
plt.title("Actual vs Predicted Water Temperature")
plt.show()

# Residual Plot
residuals = y_test - y_pred
plt.figure(figsize=(6,5))
plt.scatter(y_pred, residuals, alpha=0.5)
plt.axhline(0, color='r', linestyle='--')
plt.xlabel("Predicted Temperature")
plt.ylabel("Residual Error")
plt.title("Residual Plot")
plt.show()

# Ridge Regression
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
ridge_pred = ridge.predict(X_test)
print("Ridge Regression R2:", r2_score(y_test, ridge_pred))
```

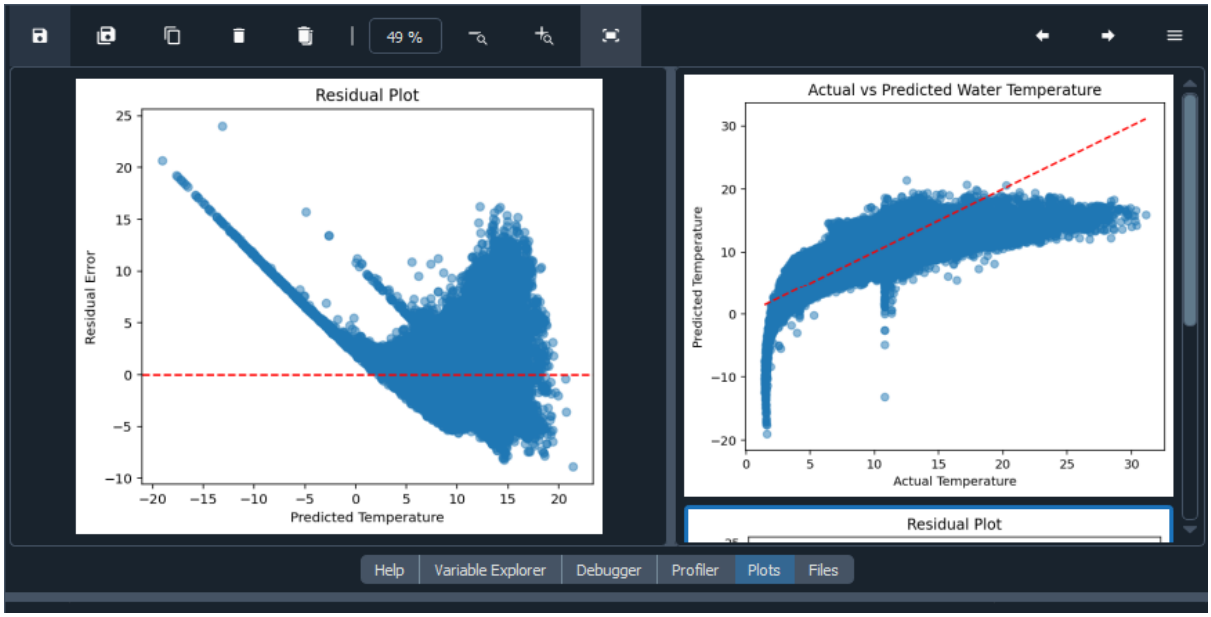
```python
# SUDHIKSHA 24BAD117
# SCENARIO 2 – MULTI-FILE VERSION

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, roc_curve, roc_auc_score
)

files = [
    "LICI - minute ata.csv",
    "LICI - 3 minute data.csv",
    "LICI - 5 minute data.csv",
    "LICI - 10 minute data.csv",
    "LICI - Daily data.csv"
]


def find_col(df, possible):
    for name in possible:
        if name in df.columns:
            return name
    return None
```

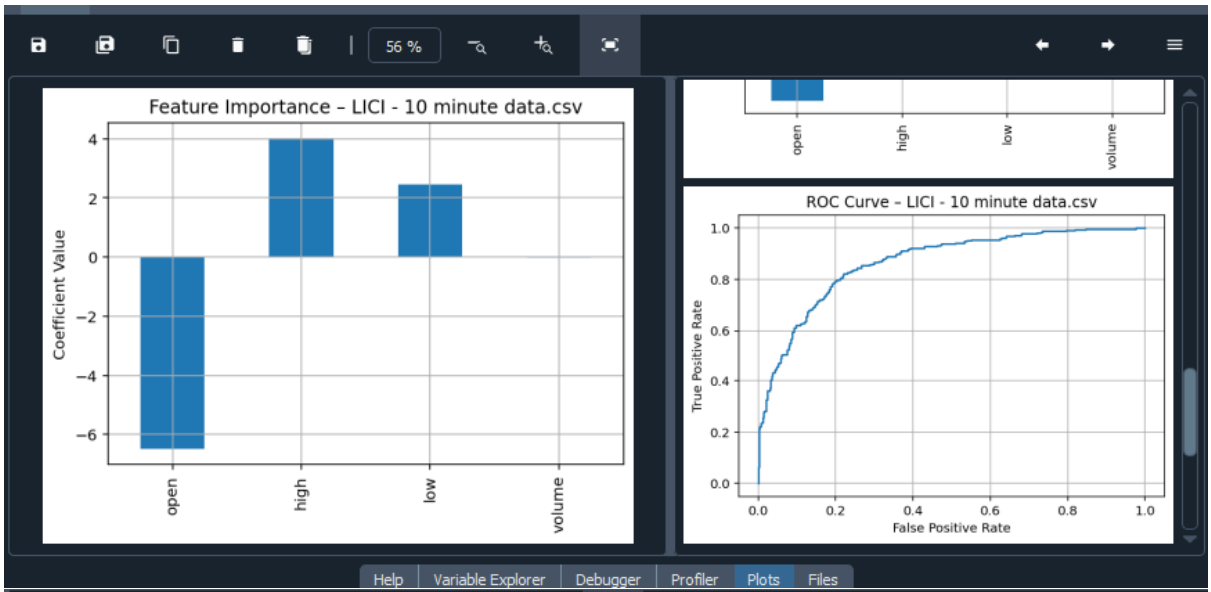
```python

for file in files:

    print("\n" + "="*50)
    print("Processing file:", file)

    df = pd.read_csv(file)
    df.columns = df.columns.str.strip().str.lower()

    open_col   = find_col(df, ['open'])
    high_col   = find_col(df, ['high'])
    low_col    = find_col(df, ['low'])
    close_col  = find_col(df, ['close', 'adj close', 'adj_close'])
    volume_col = find_col(df, ['volume'])

    if None in [open_col, high_col, low_col, close_col, volume_col]:
        print("Required columns missing → Skipping file")
        continue

    df['price_movement'] = np.where(df[close_col] > df[open_col], 1, 0)
    df = df[[open_col, high_col, low_col, volume_col, 'price_movement']].dropna()

    X = df[[open_col, high_col, low_col, volume_col]]
    y = df['price_movement']

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    X_train, X_test, y_train, y_test = train_test_split(
        X_scaled, y, test_size=0.2, random_state=42, stratify=y
    )

    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)
```

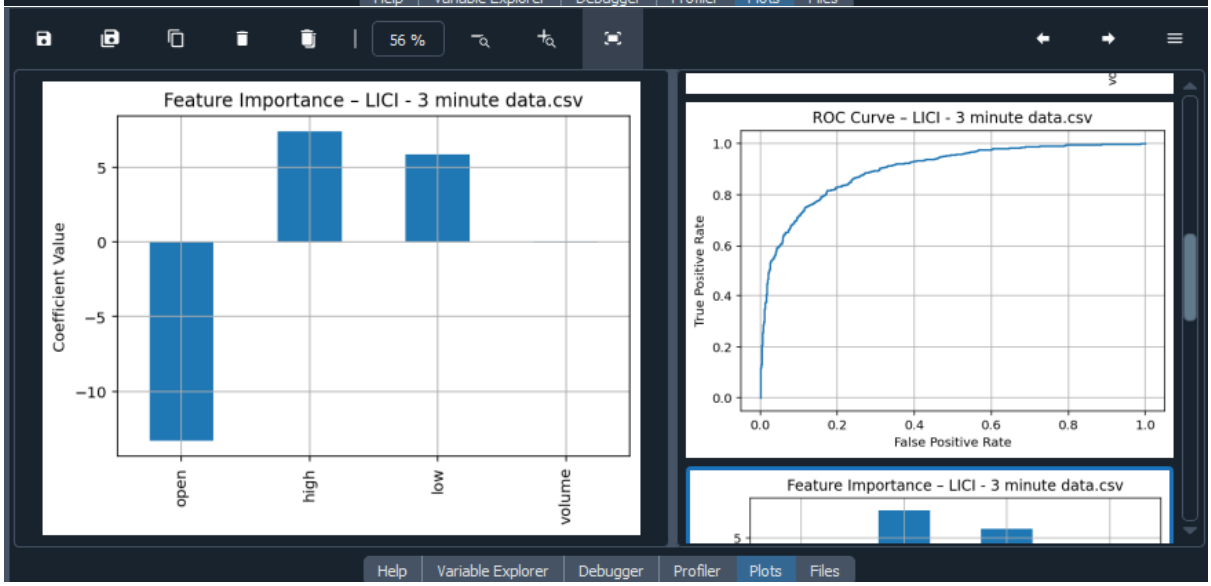
```
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]

    acc  = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec  = recall_score(y_test, y_pred)
    f1   = f1_score(y_test, y_pred)
    cm   = confusion_matrix(y_test, y_pred)
    auc  = roc_auc_score(y_test, y_prob)

    print("\n--- Performance Metrics ---")
    print("Accuracy :", acc)
    print("Precision:", prec)
    print("Recall   :", rec)
    print("F1-Score :", f1)
    print("ROC-AUC  :", auc)
    print("Confusion Matrix:\n", cm)


    # ROC Curve
    fpr, tpr, _ = roc_curve(y_test, y_prob)

    plt.figure()
    plt.plot(fpr, tpr)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title(f"ROC Curve - {file}")
    plt.grid()
    plt.show()

    # Feature Importance
    importance = pd.Series(model.coef_[0], index=X.columns)

    plt.figure()
```

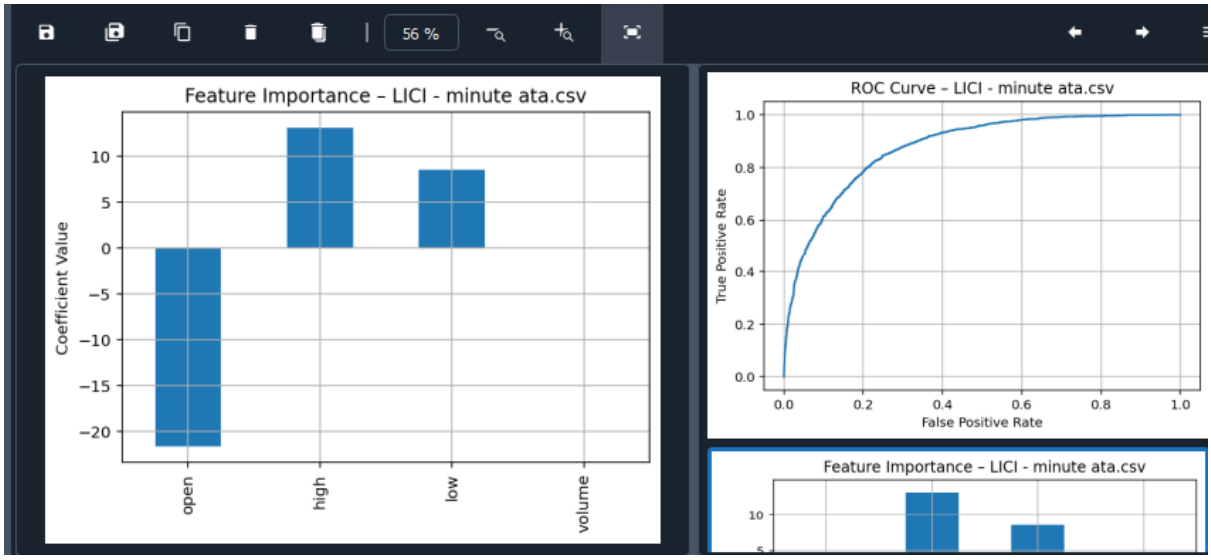
```
    # Feature Importance
    importance = pd.Series(model.coef_[0], index=X.columns)

    plt.figure()
    importance.plot(kind="bar")
    plt.title(f"Feature Importance - {file}")
    plt.ylabel("Coefficient Value")
    plt.grid()
    plt.show()


    # Hyperparameter tuning
    param_grid = {
        "C": [0.01, 0.1, 1, 10],
        "penalty": ["l2"],
        "solver": ["lbfgs"]
    }

    grid = GridSearchCV(
        LogisticRegression(max_iter=1000),
        param_grid,
        cv=5,
        scoring="accuracy"
    )

    grid.fit(X_train, y_train)

    print("\nBest Hyperparameters:", grid.best_params_)
    print("Tuned Accuracy:", accuracy_score(y_test, grid.best_estimator_.predict(X_test)))

print("\nAll files processed successfully.")
```

OUTPUT:

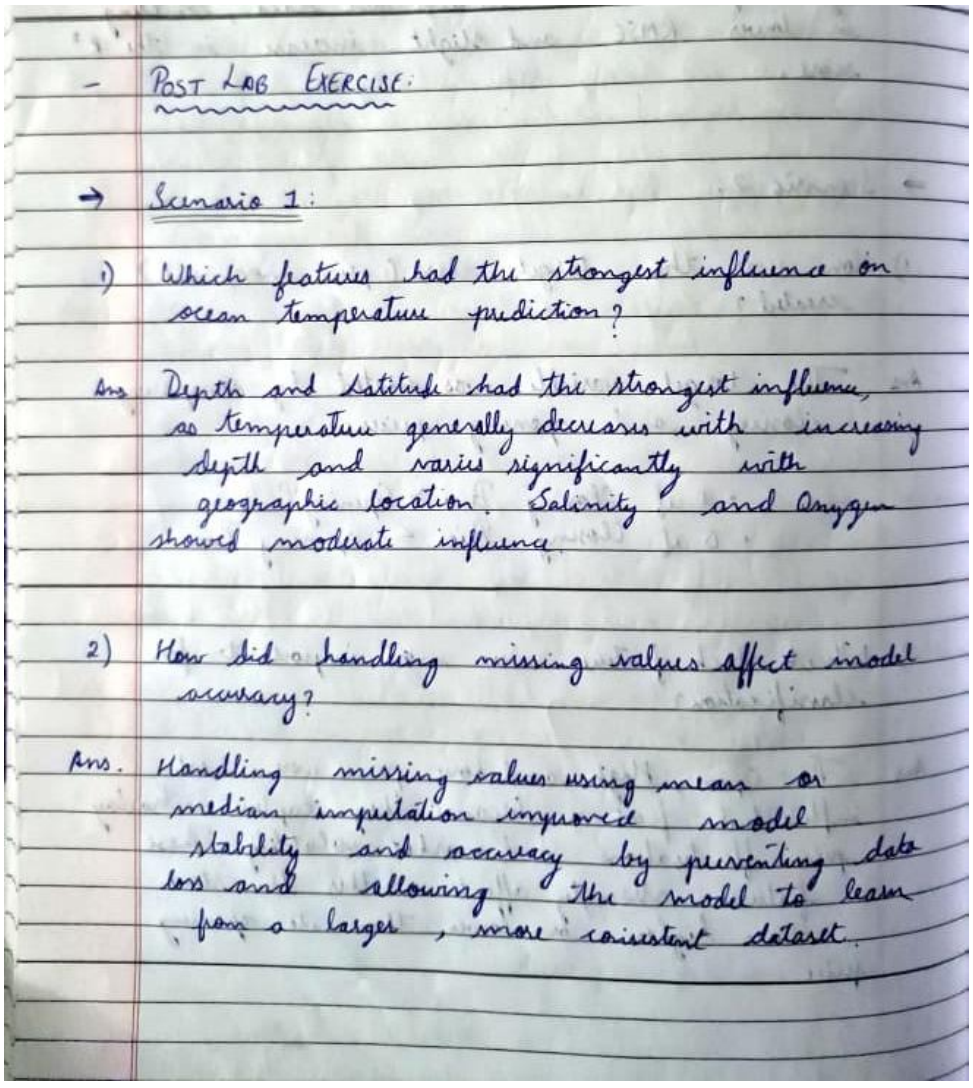

49 %
Residual Plot
Residual Error
Predicted Temperature
Actual vs Predicted Water Temperature
Predicted Temperature
Actual Temperature
Residual Plot
Help
Variable Explorer
Debugger
Profiler
Plots
Files


56 %
Feature Importance – LICI - 10 minute data.csv
Coefficient Value
open
high
low
volume
ROC Curve – LICI - 10 minute data.csv
True Positive Rate
False Positive Rate
Help
Variable Explorer
Debugger
Profiler
Plots
Files


56 %
Feature Importance – LICI - 3 minute data.csv
Coefficient Value
open
high
low
volume
ROC Curve – LICI - 3 minute data.csv
True Positive Rate
False Positive Rate
Feature Importance – LICI - 3 minute data.csv
Help
Variable Explorer
Debugger
Profiler
Plots
Files



POST-LAB QUESTIONS (PROVIDE BRIEF ANSWERS TO THE FOLLOWING QUESTIONS)



– POST LAB EXERCISE:

→ Scenario 1:

1) Which features had the strongest influence on ocean temperature prediction?

Ans. Depth and latitude had the strongest influence as temperature generally decreases with increasing depth and varies significantly with geographic location. Salinity and Oxygen showed moderate influence.

2) How did handling missing values affect model accuracy?

Ans. Handling missing values using mean or median imputation improved model stability and accuracy by preventing data loss and allowing the model to learn from a larger, more consistent dataset.

3) What improvement was observed after applying regularization?

Ans After applying regularization (Ridge / Lasso), the model showed reduced of overfitting by penalizing large coefficients. This led to better generalization on unseen test data, resulting in lower RMSE and slight increase in the R^2 score.

⇒ Scenario 2:

1) How was the target class (price movement) created?

Ans The target variable was created by comparing closing and opening prices:

- 1 if Closing Price > Opening Price
- 0 if Closing Price ≤ Opening Price

2) Which stock features were most important for classification?

Ans The Open, High and low prices were most influential features because they capture intraday price fluctuations and market volatility. These features strongly affect whether the stock closes higher or lower than its opening price.

3) How did feature scaling impact convergence and accuracy?

Ans. Feature scaling standardized all input variables to common scale, preventing features with larger numerical ranges from dominating the learning process. It improved speed of Logistic Regression algorithm, resulted in more consistent coefficient estimation.

4) What changes were observed after hyperparameter tuning?

Ans. Hyperparameter tuning, particularly adjusting the regularization strength and penalty type, improved the model's balance between bias and variance. The tuned model achieved higher accuracy and better precision.

- ASSESSMENT

Description	Max Marks	Marks Awarded
Pre Lab Exercise	5	5
In Lab Exercise	10	10
Post Lab Exercise	5	5
ViVa	10	6
Total	30	26

Faculty Signature: 6/1/26