Programming Part

1. Data Processing
   1. Loading Data:
      * Training Dataframe Shape: (6250, 12)
      * Testing Dataframe Shape: (3221, 11)
   2. Missing Values
      * Training Data:
        + Total Missing values: 924
        + Rows with Missing values: 77
      * Testing Data:
        + Total missing values: 407
        + Rows with Missing values: 3
   3. Shape after dropping rows with missing values
      * Training Data Shape after clean\_data: (6250, 12)
      * Testing Data Shape after clean\_data: (3184, 11)
   4. Features and Labels
      * Features:

NMHC(GT), C6H6(GT), PT08.S2(NMHC)m, NOx(GT), PT08.S3(NOx), NO2(GT), PT08.S4(NO2), PT08.S5(O3), T, RH, AH

* + - Label (Target): PT08.S1(CO)

1. Exploratory Data Analysis
   1. Histograms
   2. Scatter Plot
   3. Pearson’s Correlation and Heatmap
2. Linear Regression
   1. Model Initialization

Text

AI-generated content may be incorrect.

* 1. Criterion (MSE)

Schematic

AI-generated content may be incorrect.

* 1. Training loop, Loss V/s Iteration

A picture containing text

AI-generated content may be incorrect.

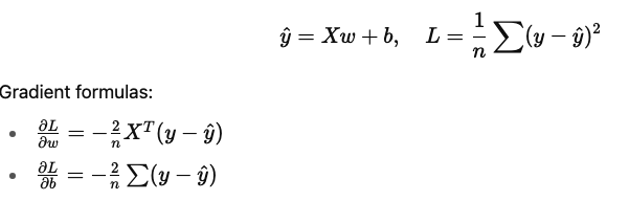
* 1. Prediction on test data

A picture containing text

AI-generated content may be incorrect.

Graphical user interface, text, application, chat or text message

AI-generated content may be incorrect.



* 1. Tune hyperparamters so that RMSE is <= 71

1. Logistic Regression
   1. Binary Label
   2. Criterion (Binary Cross Entropy)
   3. Main loop and Loss V/s Iterations
   4. Prediction on test data
2. Result Analysis – Cross Validation

The class ModelEvaluator implements the cross-validation process for the Linear Regressio and Logistic Regression classes in this code. The class method cross\_validation implements a K-Fold cross-validation approach using sklearn.model\_selection.KFold. It splits the dataset into k (in our case, 5) subsets or "folds", and performs iterative training and validation. In each iteration, one fold is used as the validation set, while the remaining k-1 folds form the training set. This ensures that every data point gets to be in the validation set exactly once, providing a robust estimate of model performance and reducing the risk of overfitting to a specific train-test split.

1. Within each fold, the method fits the given model (either a LinearRegression or LogisticRegression) on the training split and then evaluates it on the validation split. For LinearRegression, it calculates and stores the Root Mean Squared Error (RMSE) as the performance metric. For LogisticRegression, the method calculates both F1 Score and AUROC (Area Under the Receiver Operating Characteristic curve), providing a comprehensive evaluation of classification performance, especially useful for imbalanced datasets.

Finally, after completing all folds, the code aggregates the performance metrics—averaging RMSE for regression, and both F1 and AUROC for classification. These aggregated results are printed out to assess the model's generalization ability. This approach ensures that the performance results are not biased by any particular subset of data, and the variance (standard deviation) across folds gives insights into the model’s stability and consistency across different data partitions

1. .



Graphical user interface, text, application

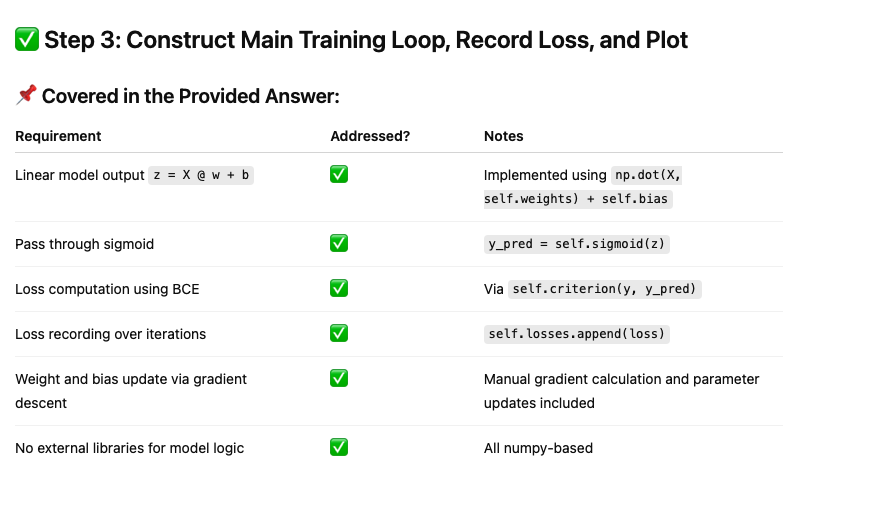
AI-generated content may be incorrect.

Logistic Regression

Step 2

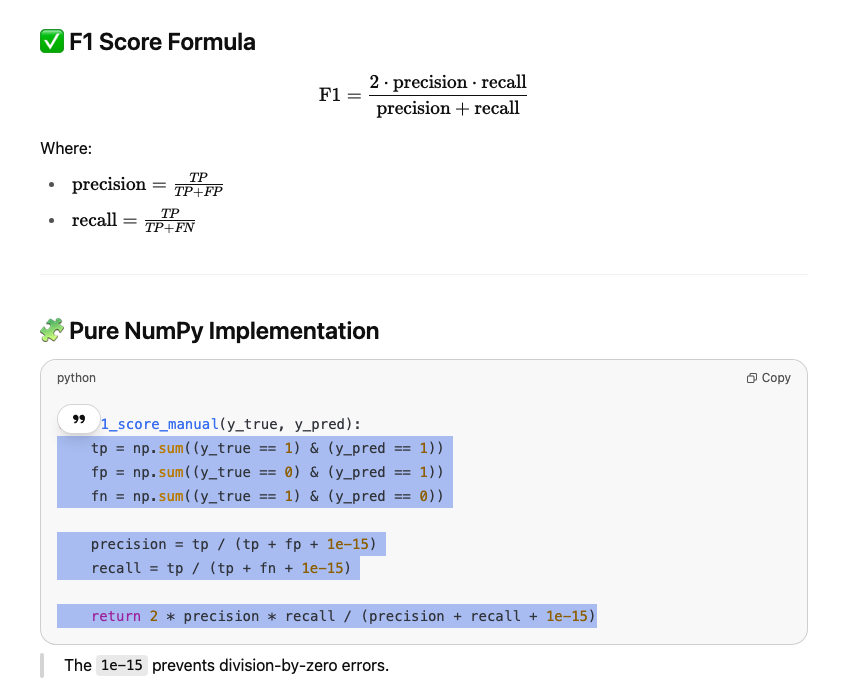
Text

AI-generated content may be incorrect.



Text, letter

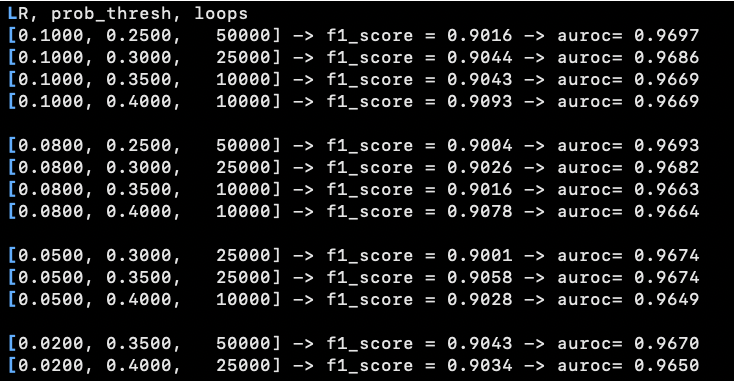
AI-generated content may be incorrect.



Graphical user interface, text, application, email

AI-generated content may be incorrect.

Hyperparameter tuning



Table

AI-generated content may be incorrect.

Graphical user interface, text, application, letter

AI-generated content may be incorrect.

During the evaluation of the Logistic Regression model, an inconsistency was observed: the F1 score consistently exceeded 0.90, while the AUROC remained significantly lower. This was unexpected, as a high F1 score typically suggests strong model performance, and AUROC should reflect that. Upon investigation, a mistake was spotted. The AUROC was being calculated using the model’s thresholded predictions (i.e., class labels 0 or 1), rather than the raw probability scores.

This distinction is critical. While the F1 score evaluates precision-recall tradeoff at a specific threshold, the AUROC measures how well the model ranks positive examples ahead of negatives across all thresholds. Supplying binary predictions to the AUROC function flattened its discriminatory power, leading to misleadingly low values. The issue was resolved by passing the predicted probabilities instead of binary class outputs to the AUROC calculation. Once corrected, the AUROC values rose appropriately—crossing the 0.90 mark—and aligned with the already strong F1 scores.

This experience underscores the importance of using appropriate prediction formats for each evaluation metric. It also highlights how misleading results can emerge if metrics are computed without full awareness of their assumptions. Ensuring the right input to each metric brought both F1 and AUROC into agreement and confirmed the robustness of the model's performance.

**[HW-1]** python Sourabh\_Pandit\_code.py

Linear Regression Results:

RMSE: 71.91679482201015

Y\_hat\_test: [1091.9516 1256.358 1142.7447 ... 826.221 1009.924 1082.277]

Logistic Regression Results:

F1-Score = 0.9011

AUROC= 0.9667

Y\_hat\_test: [0.786 0.963 0.867 ... 0.0410 0.52102 0.8191]

Cross Validation Results:

Linear Regression: RMSE: mean = 71.7718, Std Dev = 1.7759

Logistic Regression: F1: Mean = 0.9004, Std Dev = 0.0088

Logistic Regression: AUROC: Mean = 0.9663, Std Dev = 0.0033

/Users/spandit/proj/CSCE633-ML/HW-1/Sourabh\_Pandit\_code.py:563: DeprecationWarning: `trapz` is deprecated. Use `trapezoid` instead, or one of the numerical integration functions in `scipy.integrate`.

auroc = np.trapz(tprs, fprs)

Hello World!