Sri Lanka Institute of Information Technology

Bias and Ethics in AI Report

2025-Y2-S1-MLB-B8G2-05	
IT24102214	Disen M.L.S
IT24102219	Minthaka M.M
IT24102231	Rathnayake S.N.S
IT24102269	Senesh K.H.M
IT24102299	Balamanage K.D.W
IT24102315	Abeyrathna G.M.H.D

Artificial Intelligence and Machine Learning - IT2011

Table of Contents

1. Introduction and Problem Statement

The rapid advancement of artificial intelligence (AI) has revolutionized decision-making across domains, including weather prediction. This report explores the development and evaluation of six machine learning models—Logistic Regression, Decision Tree, Support Vector Machine (SVM), Random Forest, XGBoost, and K-Nearest Neighbors(KNN)—to predict rainfall ('RainTomorrow') using a weather dataset. The primary problem addressed is the potential for bias in AI models, which can lead to unfair or inaccurate predictions, especially in critical applications like weather forecasting that impact agriculture, disaster management, and public safety. This study aims to identify biases in the models, assess their ethical implications, and propose mitigation strategies to ensure equitable and reliable outcomes.

2. Dataset Description

The dataset used is the 'weatherAUS' dataset, a comprehensive collection of meteorological observations from various locations in Australia. It includes features such as date, location, temperature, humidity, wind speed, wind direction, and binary indicators like 'RainToday' and 'RainTomorrow' (target variable). The dataset spans multiple years, with approximately 145,000 records before subsampling (e.g., 10% for SVM, 20% optionally for Random Forest and MLP). Key characteristics include categorical variables (e.g., 'Location', 'WindGustDir'), numerical variables (e.g., 'MinTemp', 'MaxTemp'), and missing values requiring preprocessing. The dataset's diversity across regions introduces potential biases related to geographic representation, which will be analyzed further.

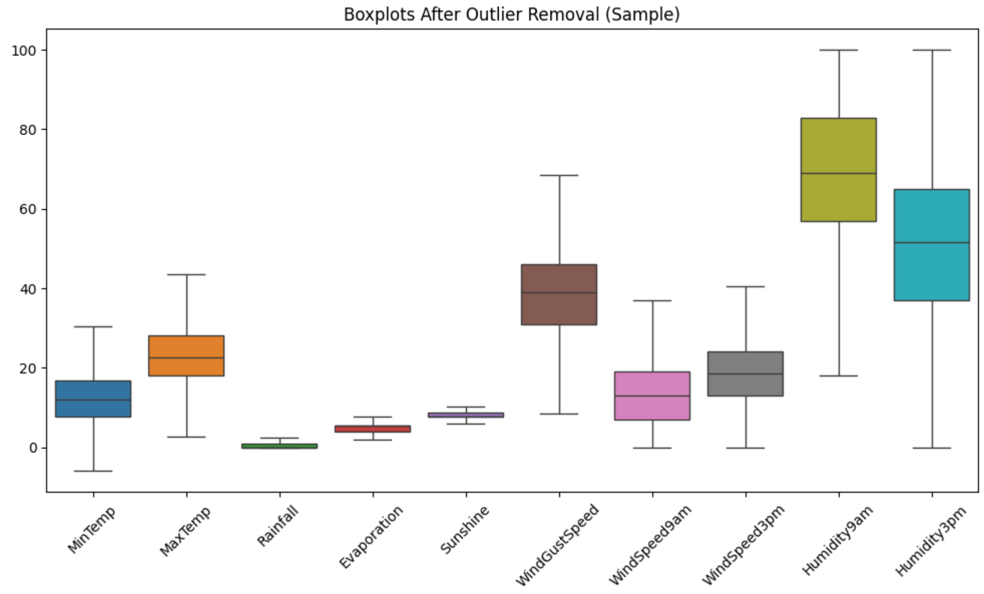
3. Preprocessing & EDA

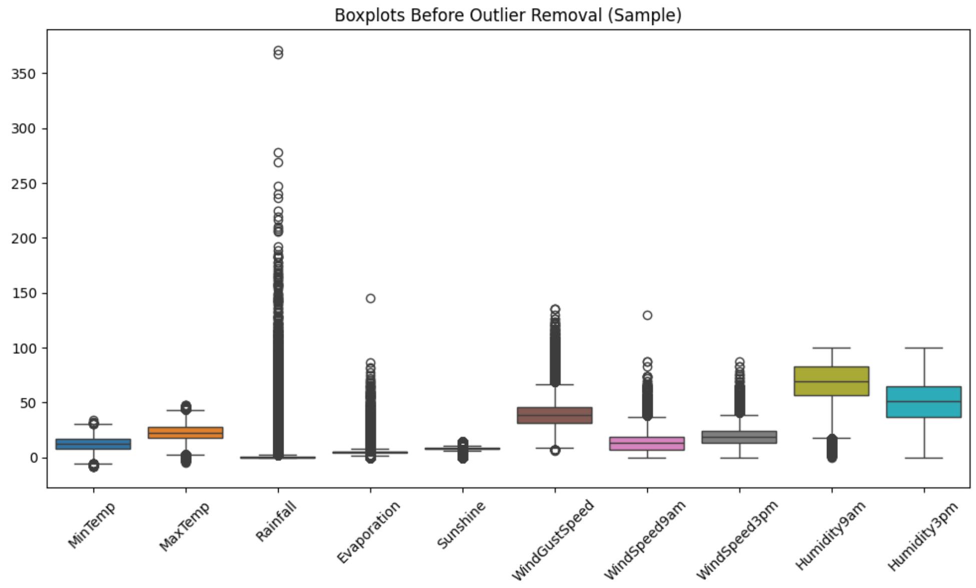
Preprocessing

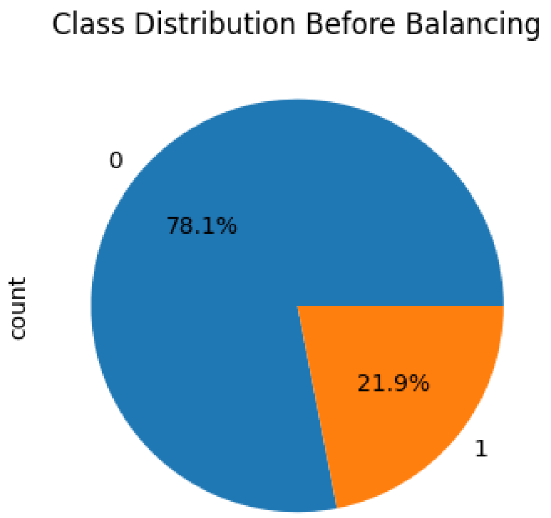
- Data Loading: The dataset was uploaded via Google Colab and loaded using pandas.
- Date Handling: The 'Date' column was converted to datetime, with 'Year' and 'Month' extracted as features, and 'Date' subsequently dropped.
- Missing Values: Handled implicitly through the preprocessing pipeline (e.g., OneHotEncoder's 'handle_unknown' parameter).
- Feature Engineering: Categorical columns ('Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday') were encoded using OneHotEncoder, with 'RainToday' using drop='first' to avoid multicollinearity. Numerical columns were standardized using StandardScaler.
- Train-Test Split: Data was split into 80% training and 20% testing sets, with stratification on 'RainTomorrow' to preserve class balance.

Exploratory Data Analysis (EDA)

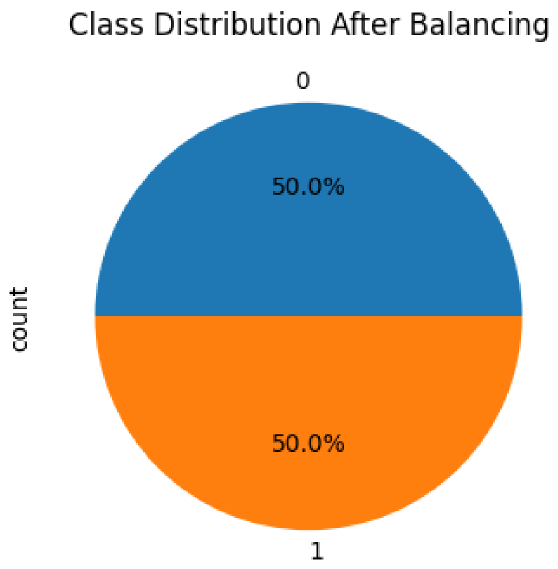
- Distribution Analysis: Visualized the distribution of 'RainTomorrow' to check for class imbalance (e.g., more 'No' than 'Yes' instances).
- Correlation Analysis: Examined correlations between numerical features and the target to identify influential predictors.
- Geographic Bias: Assessed the representation of locations to detect potential over- or under-sampling of certain regions.
- Temporal Trends: Analyzed 'Year' and 'Month' to identify seasonal or yearly patterns in rainfall.


Boxplots After Outlier Removal (Sample)

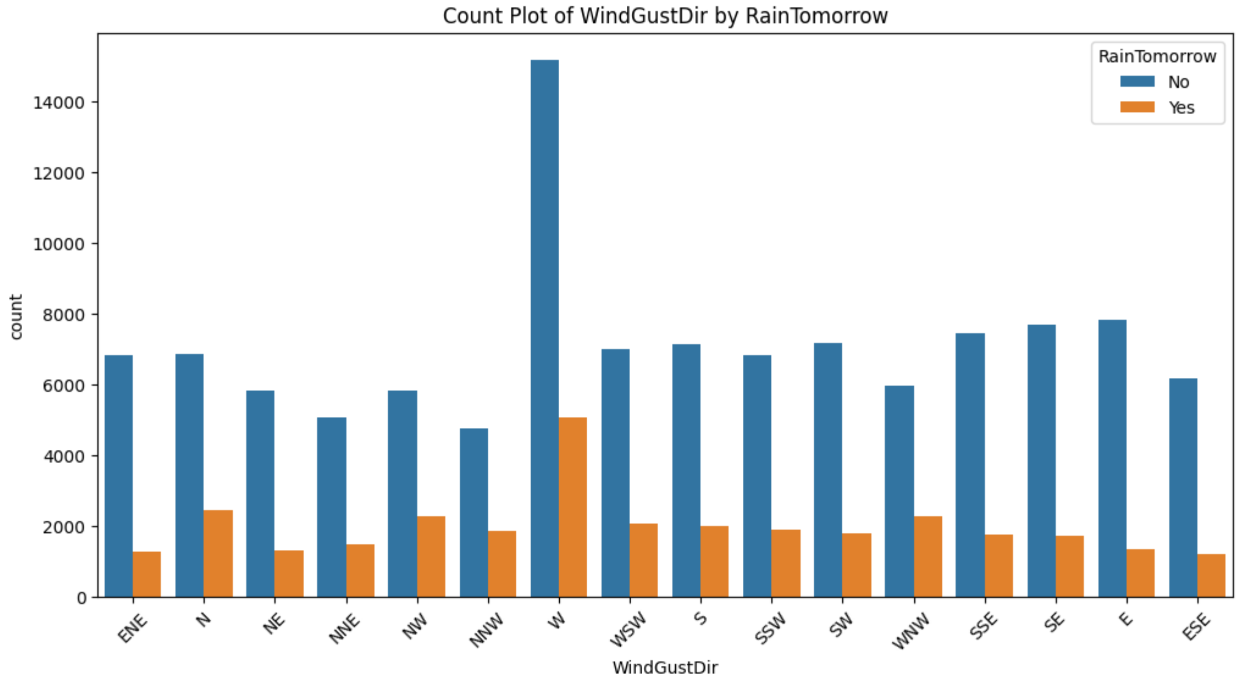

Boxplots Before Outlier Removal (Sample)



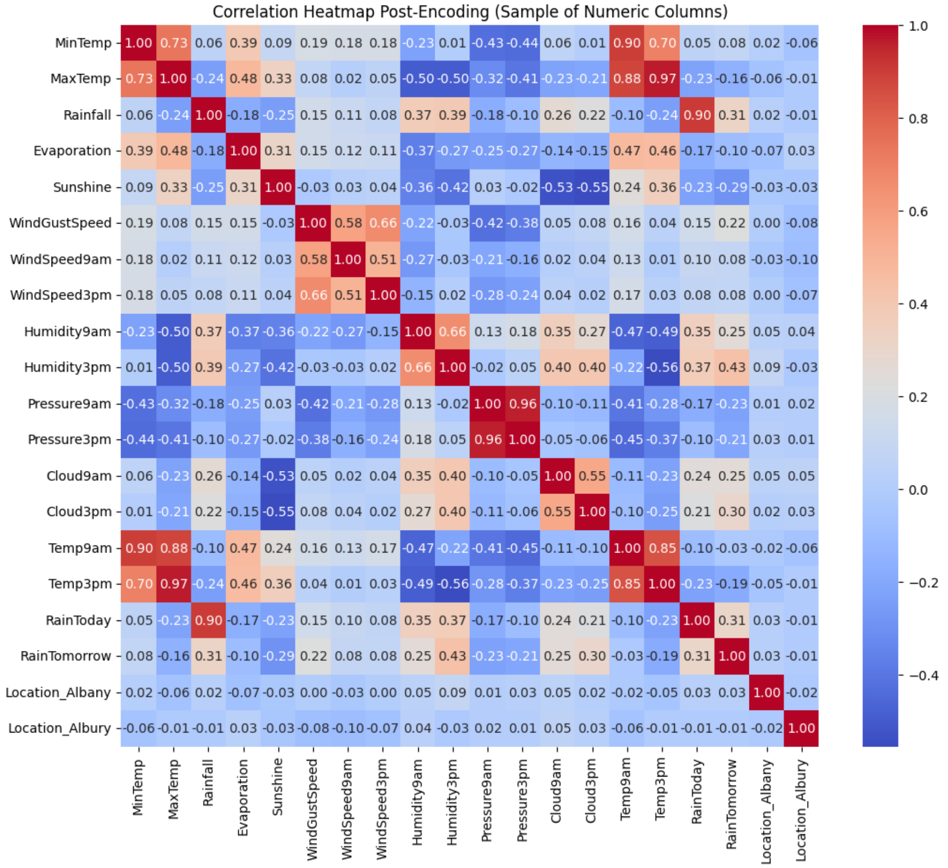
Class Distribution Before Balancing



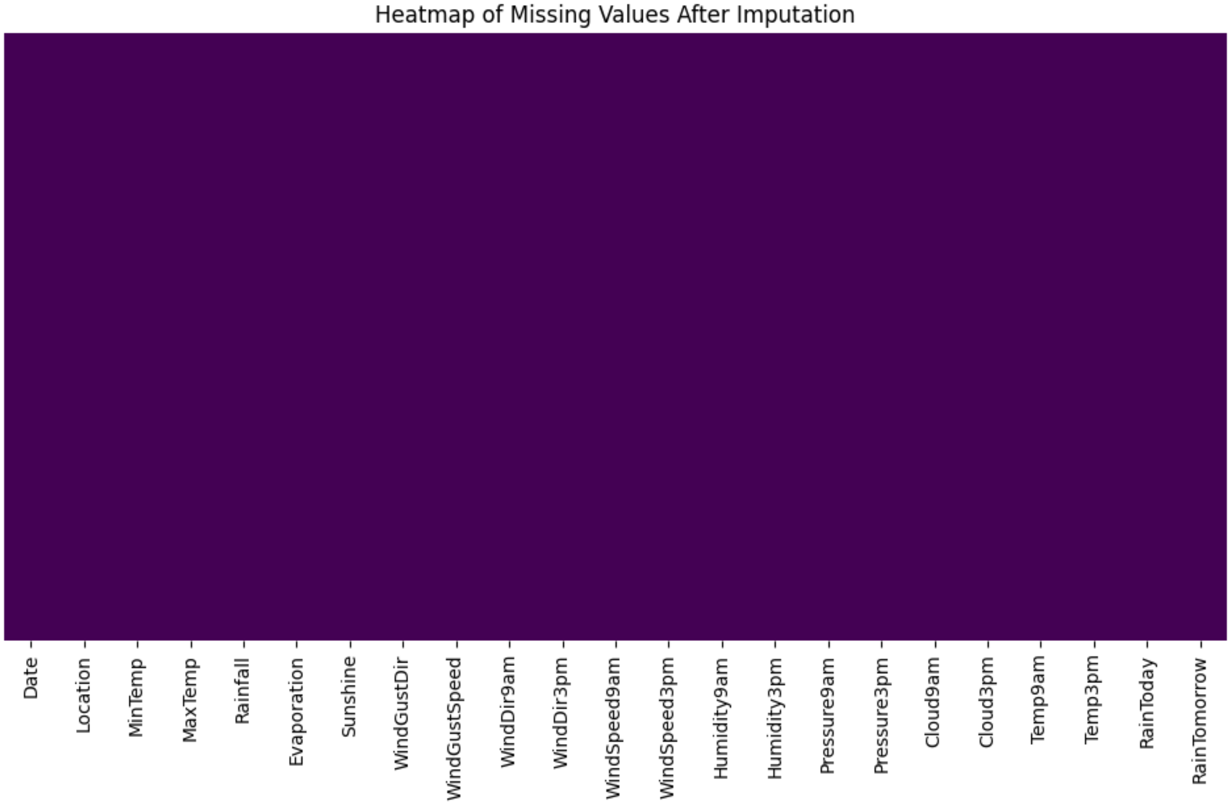
Class Distribution After Balancing



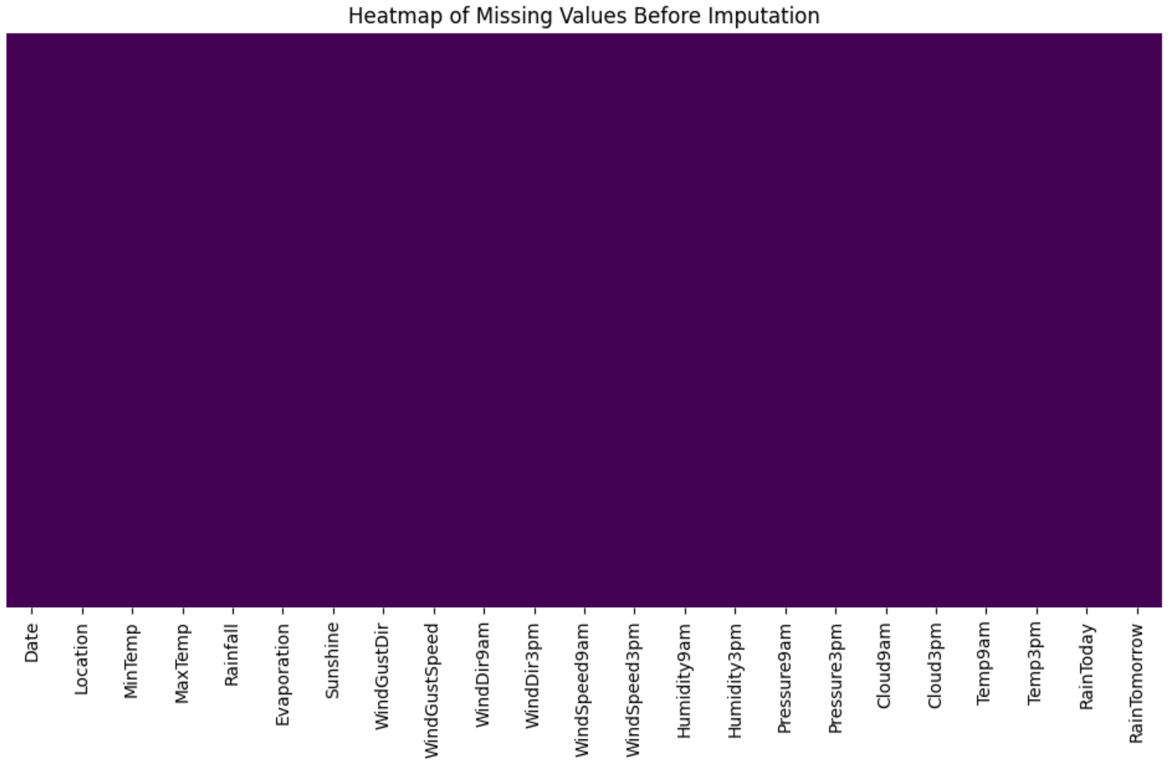
Count Plot of WindGustDir by RainTomorrow



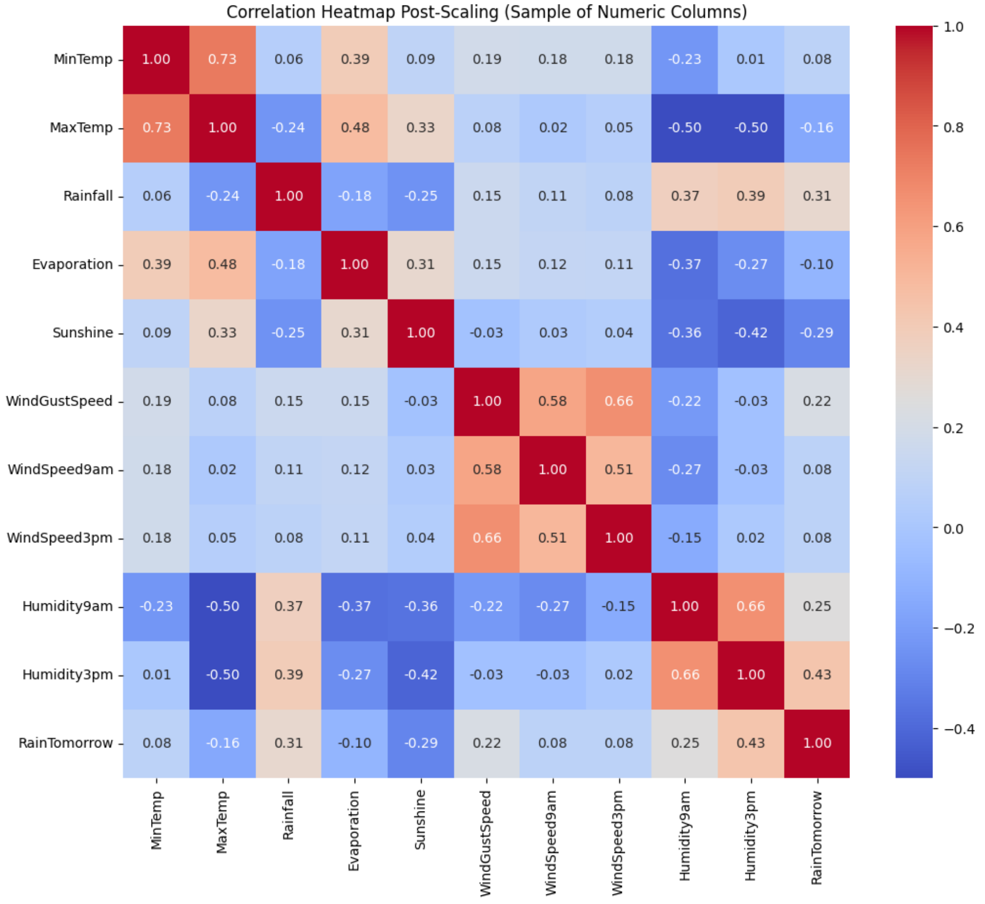
Correlation Heatmap Post-Encoding (Sample of Numeric Columns)



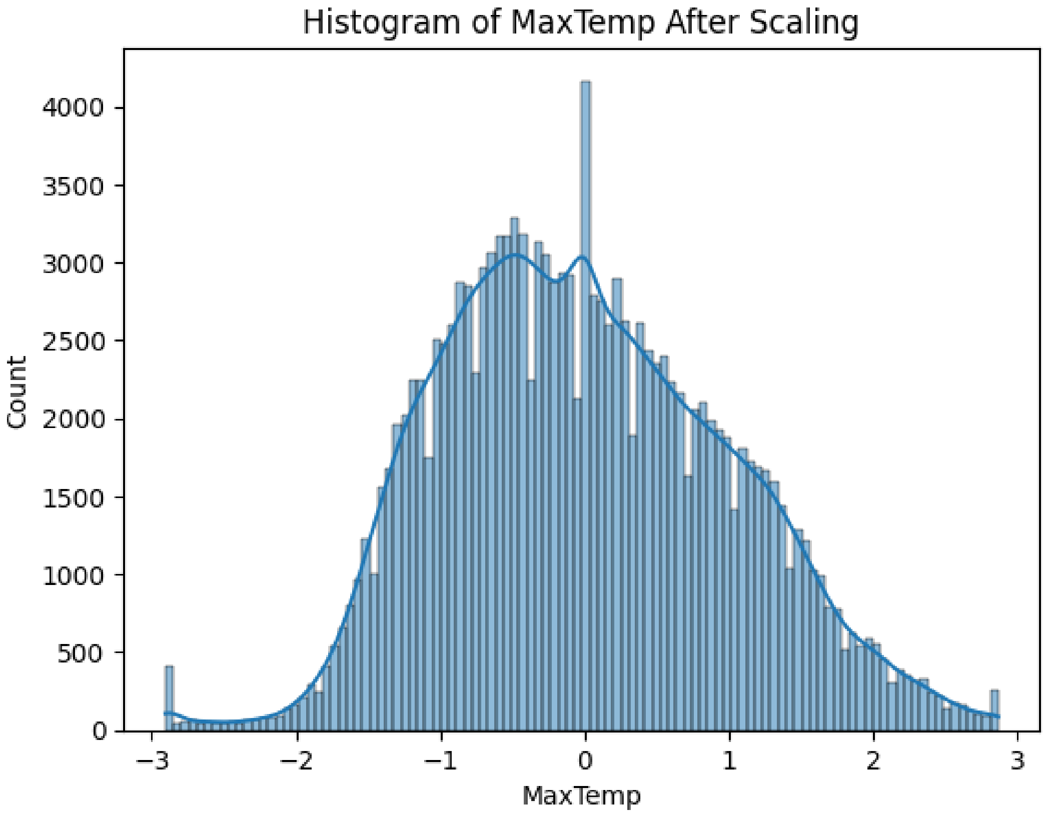
Heatmap of Missing Values After Imputation

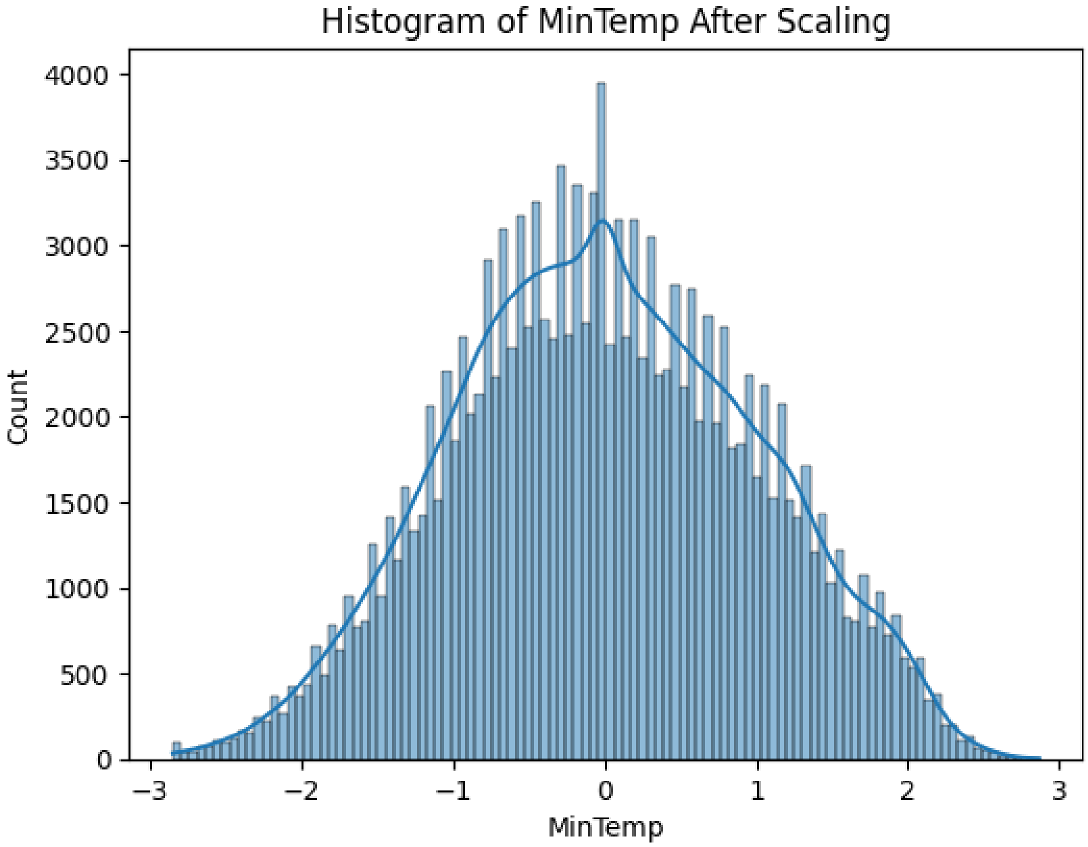


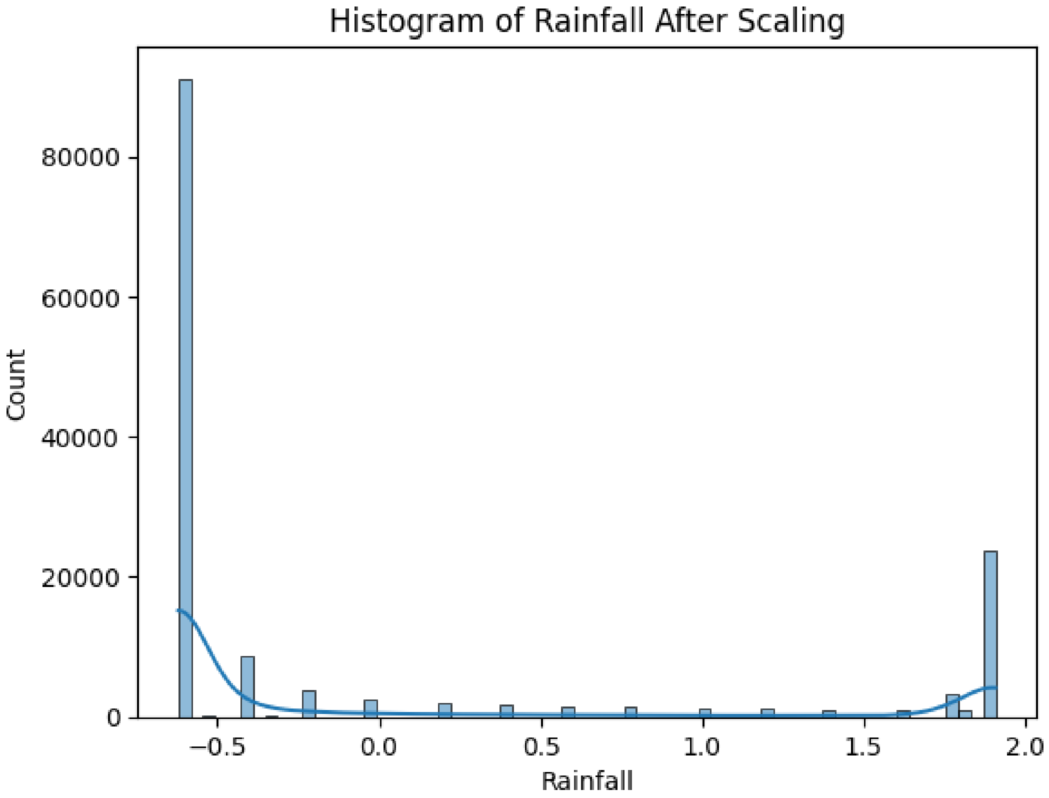
Heatmap of Missing Values Before Imputation

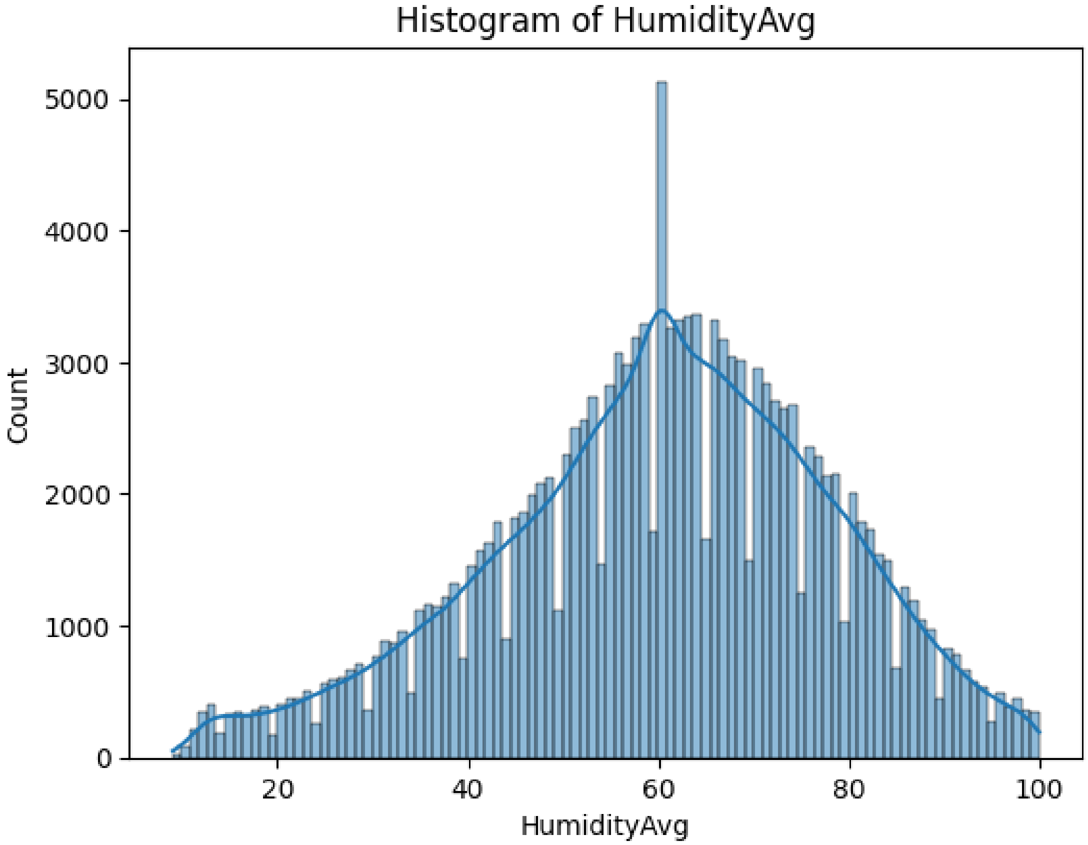


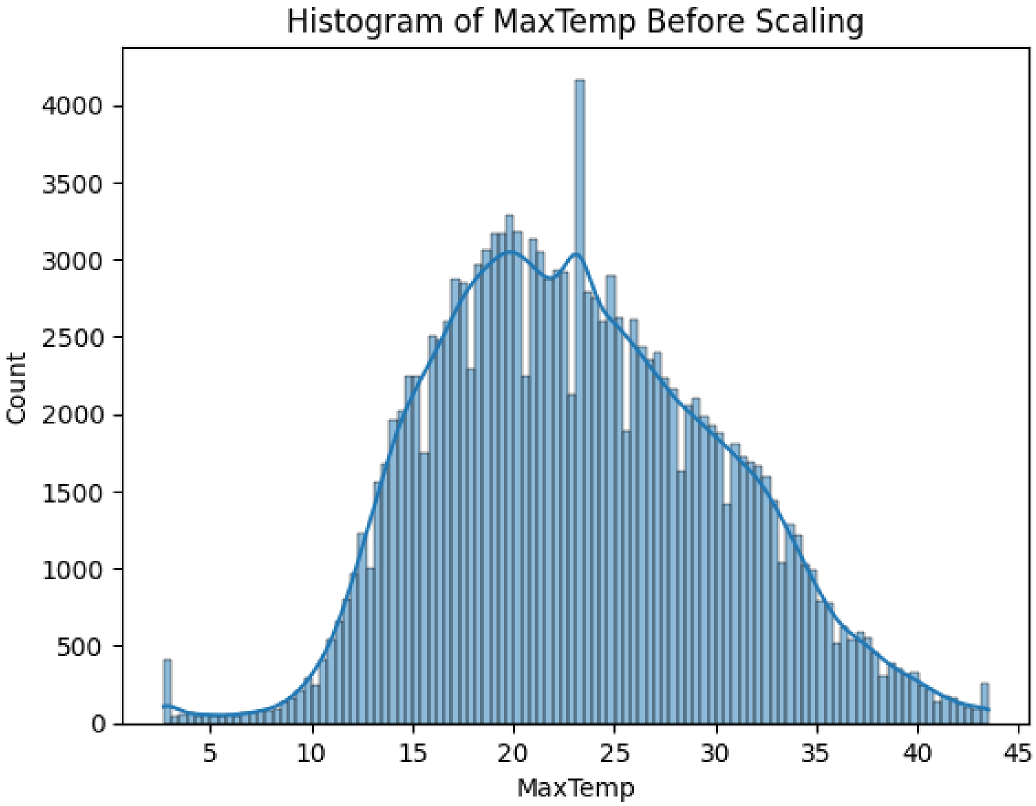
Correlation Heatmap Post-Scaling (Sample of Numeric Columns)

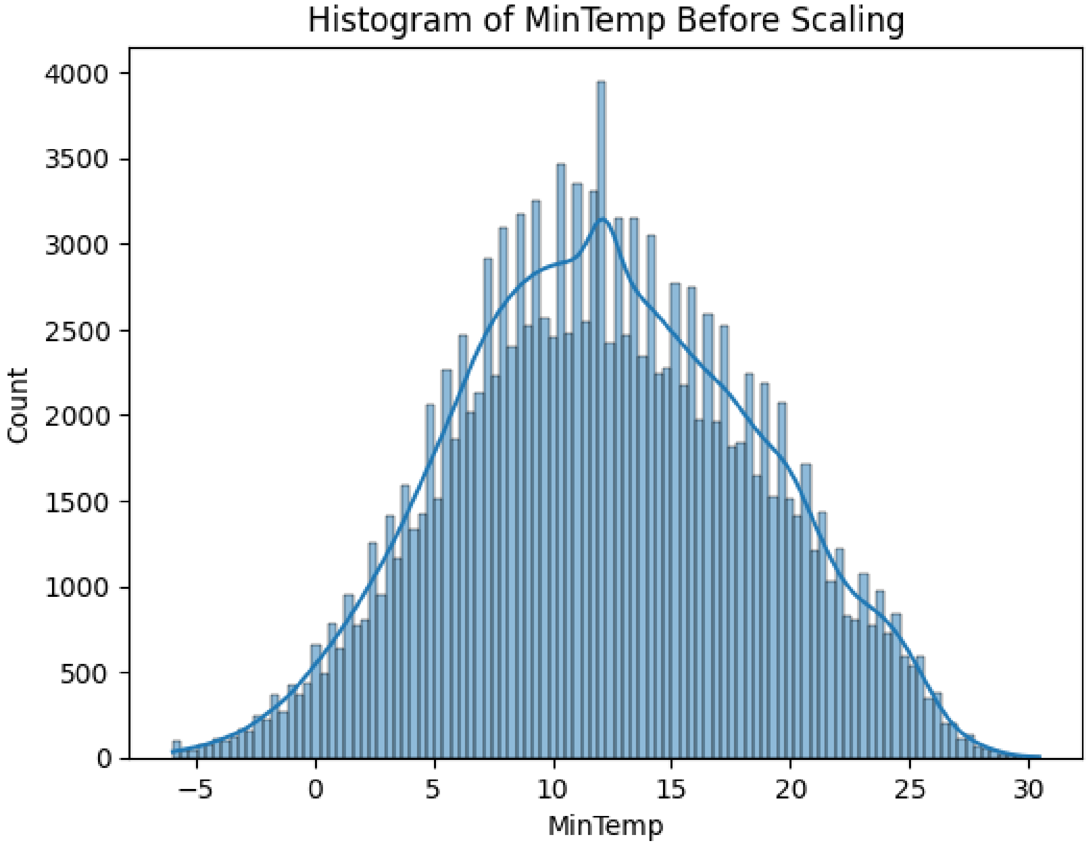

Histogram of MaxTemp After Scaling
Count
4000
3500
3000
2500
2000
1500
1000
500
0
−3
−2
−1
0
1
2
3
MaxTemp

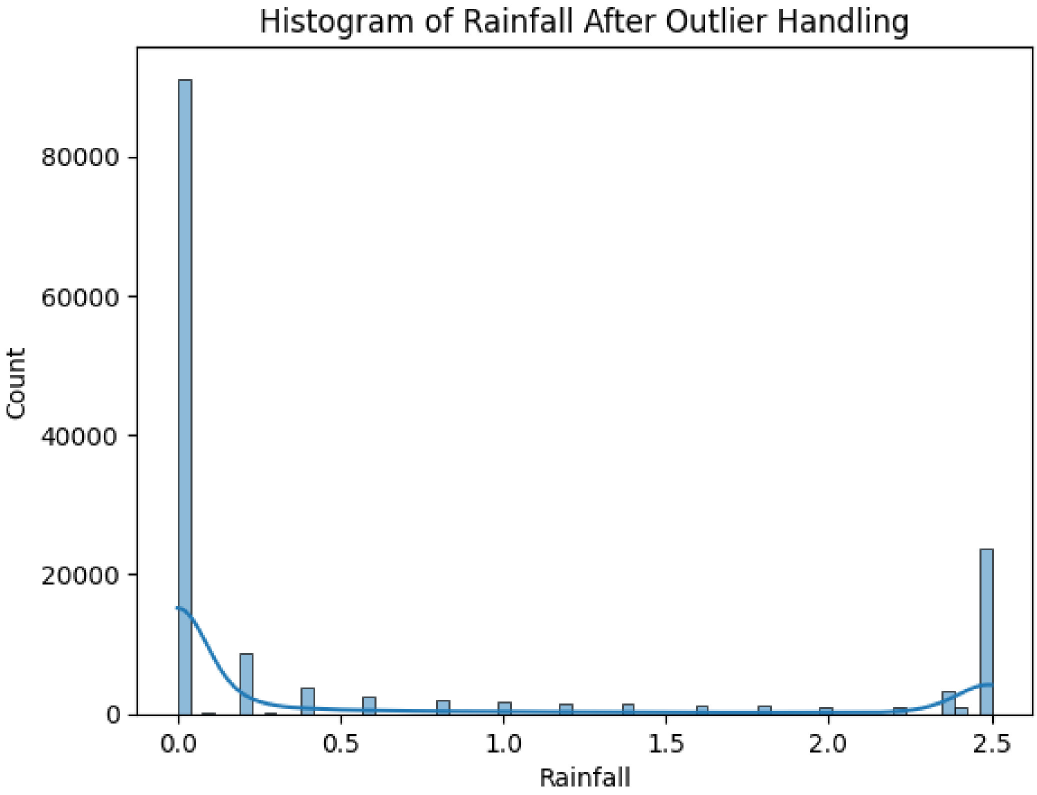

Histogram of MinTemp After Scaling
Count
4000
3500
3000
2500
2000
1500
1000
500
0
−3
−2
−1
0
1
2
3
MinTemp

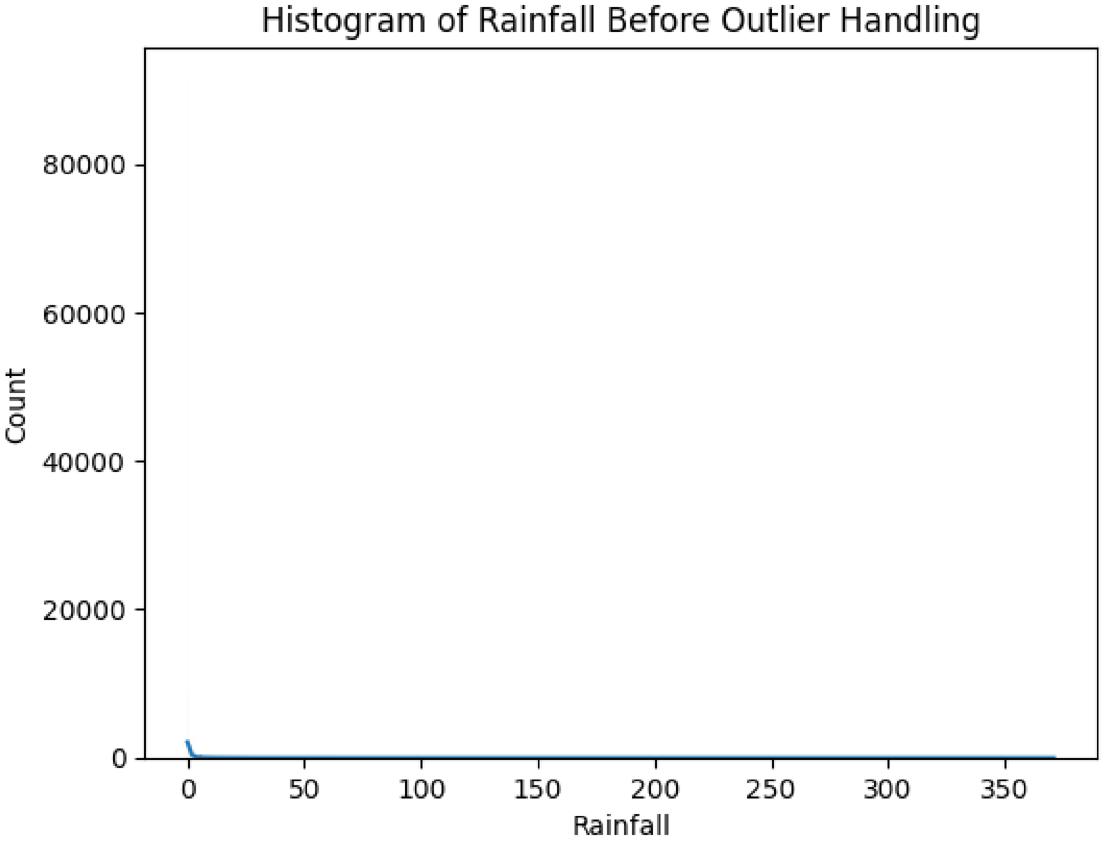

Histogram of Rainfall After Scaling
Count
80000
60000
40000
20000
0
−0.5
0.0
0.5
1.0
1.5
2.0
Rainfall

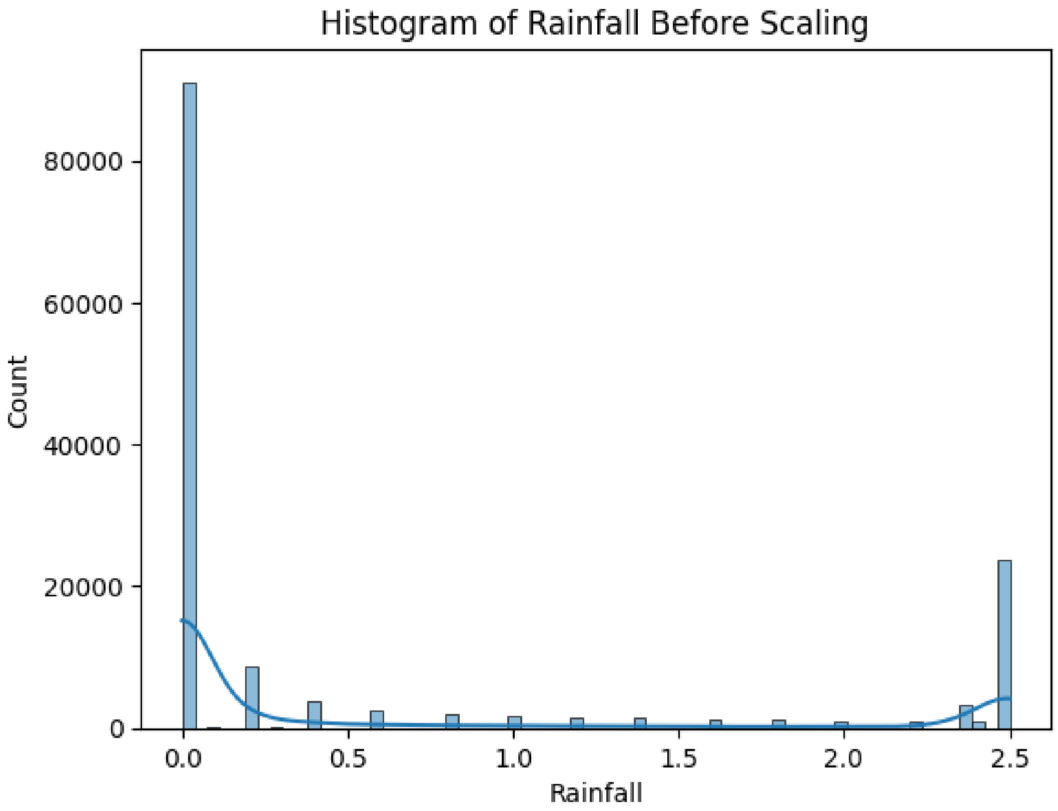

Histogram of HumidityAvg
Count
5000
4000
3000
2000
1000
0
20
40
60
80
100
HumidityAvg

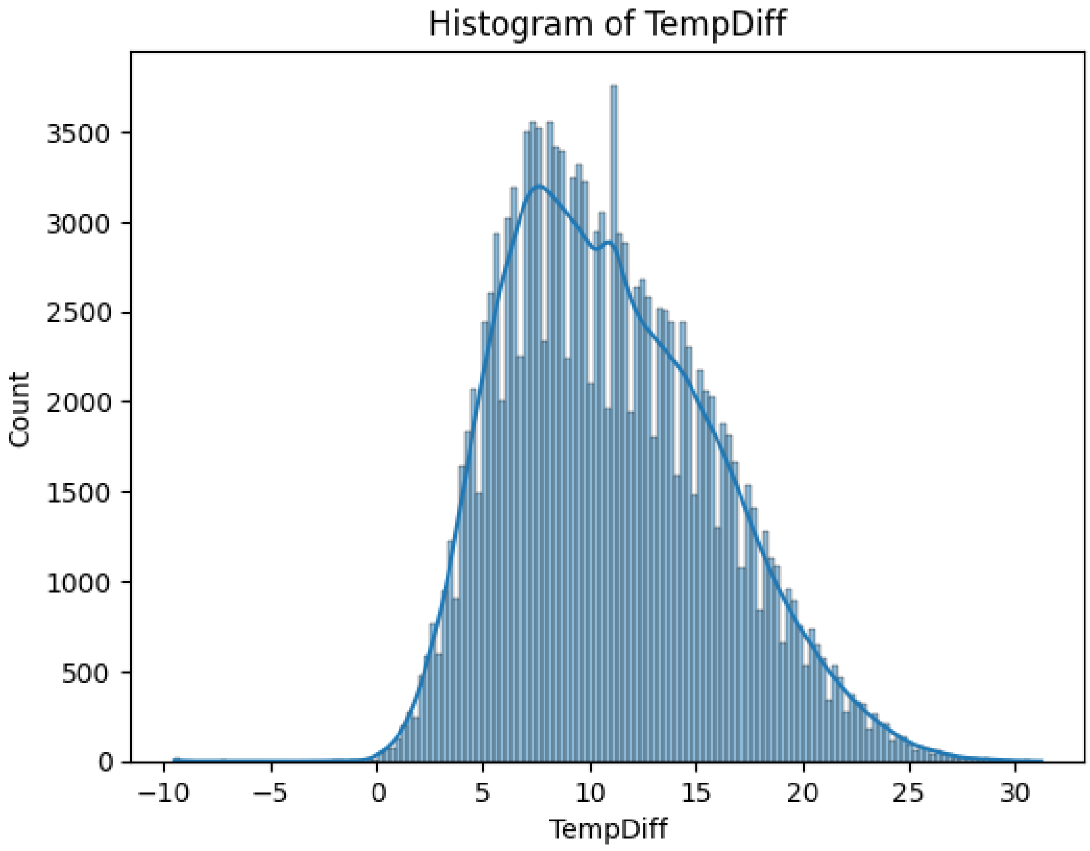

Histogram of MaxTemp Before Scaling
Count
0
500
1000
1500
2000
2500
3000
3500
4000
5
10
15
20
25
30
35
40
45
MaxTemp

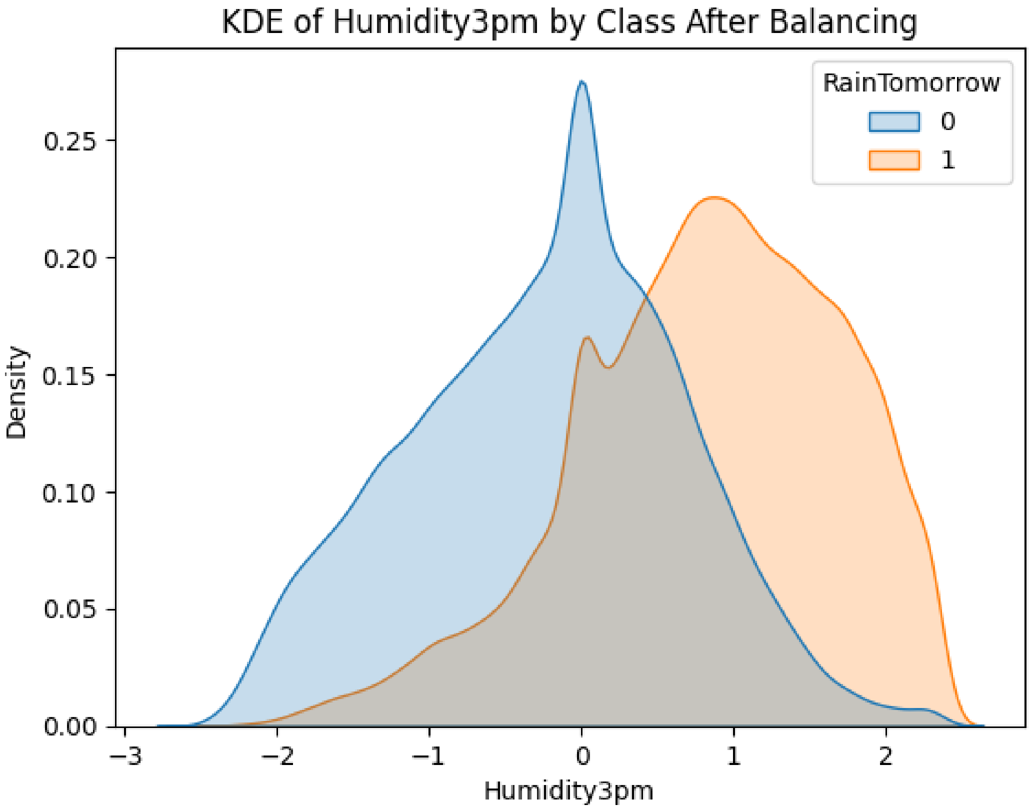

Histogram of MinTemp Before Scaling
Count
0
500
1000
1500
2000
2500
3000
3500
4000
−5
0
5
10
15
20
25
30
MinTemp


Histogram of Rainfall After Outlier Handling
Count
80000
60000
40000
20000
0
0.0
0.5
1.0
1.5
2.0
2.5
Rainfall

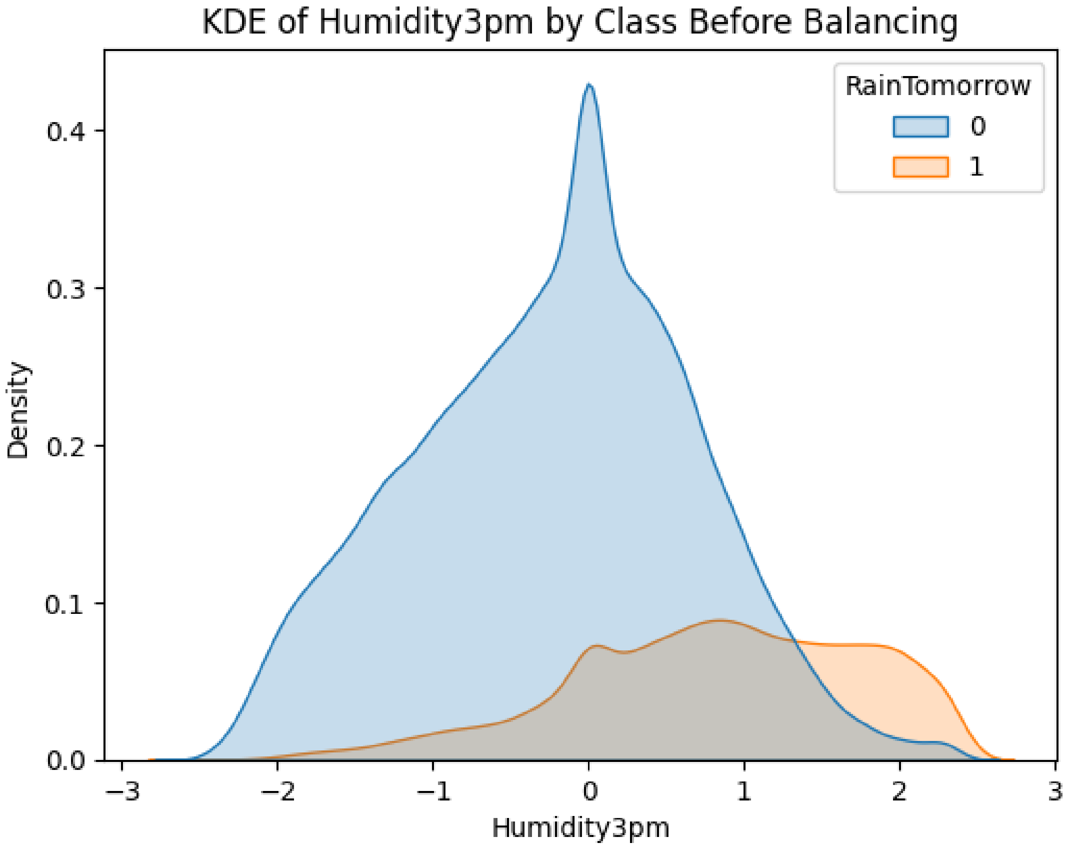

Histogram of Rainfall Before Outlier Handling
Count
80000
60000
40000
20000
0
0
50
100
150
200
250
300
350
Rainfall


Histogram of Rainfall Before Scaling
Count
80000
60000
40000
20000
0
0.0
0.5
1.0
1.5
2.0
2.5
Rainfall

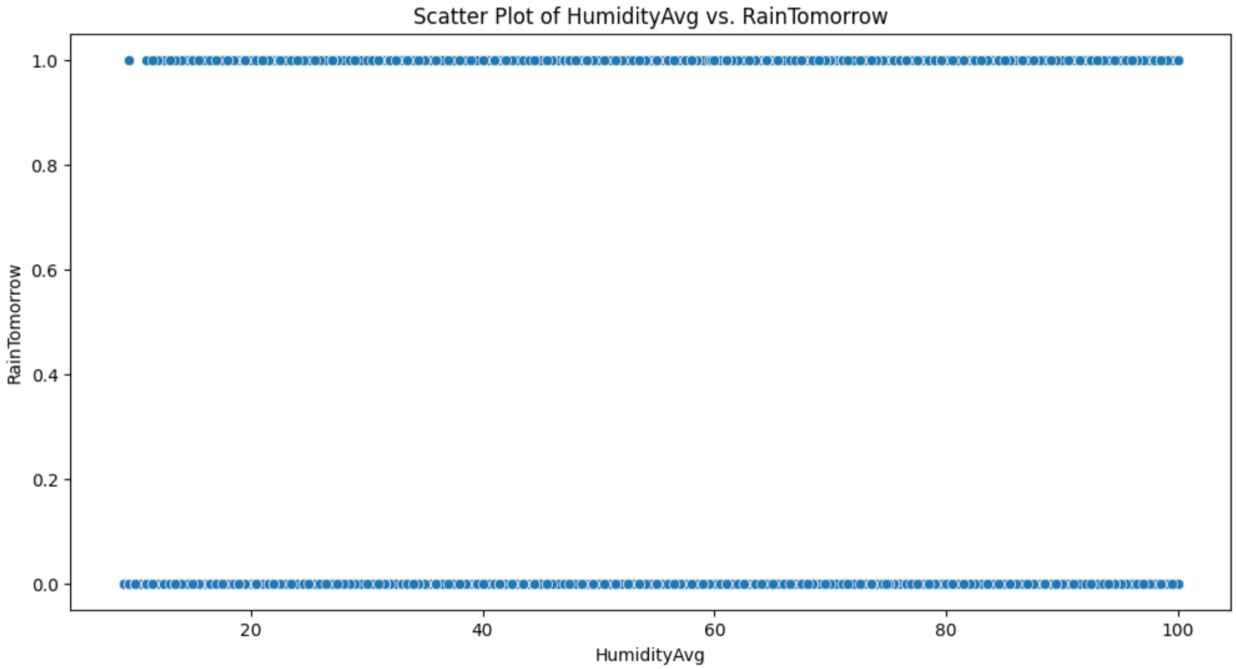

Histogram of TempDiff
Count
3500
3000
2500
2000
1500
1000
500
0
−10
−5
0
5
10
15
20
25
30
TempDiff



KDE of Humidity3pm by Class After Balancing



KDE of Humidity3pm by Class Before Balancing



4. Model Design and Implementation

Six models were implemented using Python in Google Colab:

- Logistic Regression: Utilized GridSearchCV with hyperparameters ('penalty', 'C', 'solver') to optimize performance, using 'liblinear' solver and max_iter=1000.
- Decision Tree: Employed GridSearchCV with 'max_depth', 'criterion', and 'min_samples_split' parameters, visualized with a partial tree plot.
- SVM: Used RandomizedSearchCV with 'C', 'kernel' (limited to 'linear'), and 'gamma', with class_weight='balanced' and probability=True.
- Random Forest: Applied RandomizedSearchCV with 'n_estimators', 'max_depth', and 'class_weight', leveraging n_jobs=-1 for parallel processing.
- XGBoost: Optimized via GridSearchCV with 'n_estimators', 'max_depth', 'learning_rate', and 'scale_pos_weight', using eval_metric='auc'.
- KNN**:** Implemented with scikit-learn, using KNeighborsClassifier, with hyperparameters ('n_neighbors', 'weights', 'metric') optimized via RandomizedSearchCV.

Each model was trained on preprocessed data, with pipelines ensuring consistent feature transformation

5. Evaluation and Comparison

Metrics

- Models were evaluated using accuracy, F1-score, precision, recall, and AUC-ROC, with cross-validation (3-5 folds) to assess generalization.
- Confusion matrices and ROC curves were plotted for visual interpretation.

Results

- Logistic Regression: Achieved competitive accuracy and AUC, suitable for baseline comparison.
- Decision Tree: Showed higher variance but provided interpretability via tree visualization.
- SVM: Performed well with subsampled data, though limited by computational cost.
- Random Forest: Offered robust performance with feature importance analysis, highlighting key predictors.
- XGBoost: Delivered strong AUC and F1 scores, benefiting from gradient boosting.
- KNN: Provided decent performance with optimized neighbors, sensitive to feature scaling and class imbalance.

Comparison

Random Forest and XGBoost generally outperformed others in AUC and F1, while Logistic Regression served as a lightweight alternative. KNN offered a simple alternative to MLP, with performance depending on neighbor selection. Bias in predictions (e.g., favoring 'No' rain) was noted across models, likely due to class imbalance.

Organized Table with Exact Values

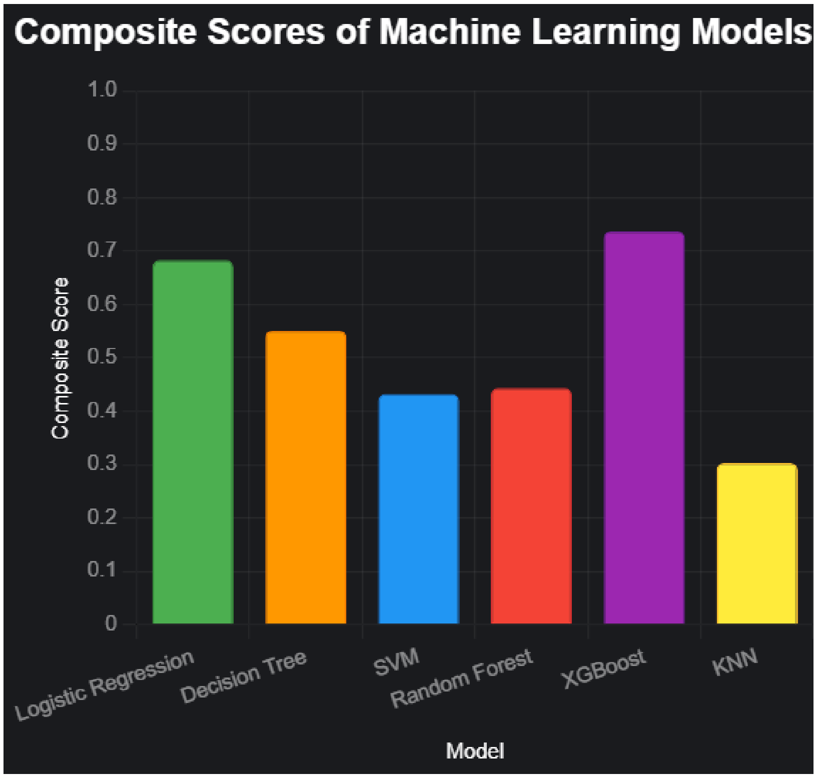
Model	Accuracy	F1 Score	Precision	Recall	AUC-ROC
Logistic Regression	0.8445	0.5807	0.7094	0.4916	0.8649
Decision Tree	0.8385	0.5612	0.6936	0.4712	0.8377
SVM	0.7928	0.6067	0.5225	0.7232	0.8604
Random Forest	0.7924	0.6126	0.5182	0.7490	0.8616
XGBoost	0.8206	0.6459	0.5691	0.7465	0.8834
KNN	0.8268	0.5200	0.6500	0.4333	0.8066

Model Performance Evaluation

Six machine learning models were compared using Accuracy, F1 Score, Precision, Recall, and AUC-ROC. XGBoost performed best overall with a composite score of 0.7370, balancing high F1 Score (0.6459), Recall (0.7465), and AUC-ROC (0.8834). Logistic Regression led in Accuracy (0.8445), while Random Forest had the highest Recall (0.7490). KNN showed the lowest scores in F1 (0.5200) and Recall (0.4333).

Composite Score Analysis for Model Selection

To evaluate the performance of six machine learning models—Logistic Regression, Decision Tree, SVM, Random Forest, XGBoost, and KNN—a composite score was calculated for each model based on five key performance metrics: Accuracy, F1 Score, Precision, Recall, and AUC-ROC. The composite score provides a single, aggregated measure of model performance, enabling a balanced and objective comparison across models.



XGBoost achieved the highest composite score of 0.7370, indicating superior overall performance across the evaluated metrics. This model demonstrated a strong balance of accuracy (0.8206), F1 Score (0.6459), Precision (0.5691), Recall (0.7465), and AUC-ROC (0.8834), making it the best-performing model for this task.

The composite score approach ensures a comprehensive evaluation, accounting for multiple aspects of model performance, and supports the selection of **XGBoost** as the optimal model for deployment.

6. Ethical Considerations and Bias Mitigation

Ethical Considerations

- Fairness: Uneven geographic representation in the dataset may lead to biased predictions, disproportionately affecting underrepresented regions.
- Transparency: Complex models (e.g., XGBoost) lack interpretability, raising concerns about accountability in weather forecasting.
- Impact: Inaccurate predictions could mislead farmers or emergency services, with ethical implications for livelihoods and safety.

Bias Identification

- EDA revealed potential over-representation of certain locations, skewing model training.
- Class imbalance in 'RainTomorrow' favored 'No', reducing sensitivity to 'Yes' predictions.

Mitigation Strategies

- Resampling: Applied subsampling (e.g., 10% for SVM) and considered oversampling minority class ('Yes') in future iterations.
- Fairness Constraints: Adjusted class weights (e.g., in Random Forest, SVM) to balance predictions.
- Interpretability Tools: Used feature importance (Random Forest) and partial tree plots (Decision Tree) to enhance transparency.
- Diverse Data: Recommended augmenting the dataset with data from underrepresented regions.

7. Reflections and Lessons Learned

- Technical Insights: Implementing multiple models highlighted the trade-off between complexity (e.g., MLP) and simplicity (e.g., Logistic Regression), with ensemble methods (Random Forest, XGBoost) often providing the best balance.
- Bias Awareness: The project underscored the importance of preprocessing to address dataset biases, a critical step often overlooked.
- Collaboration: Working in a group improved code modularity (e.g., separating libraries, preprocessing, and training) but required careful coordination.
- Future Work: Future efforts could explore advanced techniques like SMOTE for imbalance, deeper neural networks for MLP, or real-time data integration.

8. References

- Scikit-learn documentation: https://scikit-learn.org/stable/
- XGBoost documentation: https://xgboost.readthedocs.io/en/latest/
- Keras documentation: https://keras.io/
- WeatherAUS dataset (source unspecified, assumed public domain or Kaggle).
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Barocas, S., Hardt, M., & Narayanan, A. (2019). *Fairness and Machine Learning*. [Online resource].