Forecasting Rainfall : A Machine Learning Approach

Ahmed Raza Zaidi
BS(Data Science)
FAST NUCES, Lahore

I. INTRODUCTION

Rainfall prediction is a crucial element in meteorology and environmental science, influencing various sectors such as agriculture, hydrology, and disaster management. Accurate forecasting of rainfall patterns enables proactive decision-making, resource management, and disaster preparedness. Traditional methods of weather prediction have been fundamental, yet the integration of machine learning techniques offers opportunities to improve prediction accuracy and timeliness.

The dataset utilized in this project focuses on historical rainfall data collected from various meteorological stations. The dataset includes essential meteorological parameters such as temperature, humidity, wind speed, atmospheric pressure, and geographical location. These features serve as vital indicators for understanding the intricate dynamics influencing rainfall occurrences. By leveraging machine learning models, we aim to develop robust predictive models capable of accurately forecasting rainfall patterns.

A. Motivation

The motivation behind this project stems from the desire to delve deep into the realm of meteorological data analysis and predictive modeling using machine learning techniques. Our focus is on leveraging a specific dataset that encapsulates crucial meteorological parameters essential for rainfall prediction. The dataset comprises historical rainfall data collected from various meteorological stations, along with associated meteorological features such as temperature, humidity, wind speed, atmospheric pressure, and geographical coordinates. By utilizing this rich dataset, we aim to gain valuable insights into the complex dynamics influencing rainfall patterns. Rainfall prediction is of paramount importance across various domains including agriculture, hydrology, urban planning, and disaster management. Accurate forecasting of rainfall not only aids in agricultural planning and water resource management but also plays a crucial role in mitigating natural disasters such as floods and landslides.

B. Data set Description

The weather dataset used in this project encompasses a wide range of meteorological parameters and their variations over time. Each entry in the dataset represents a day's worth of weather observations collected from different locations across Australia. The

dataset comprises several key features that provide insights into daily weather patterns and help in predicting rainfall occurrences for the following day.

It captures critical metrics such as minimum and maximum temperatures (MinTemp, MaxTemp), rainfall amount (Rainfall), water evaporation (Evaporation), sunshine duration (Sunshine), wind characteristics including gust direction and speed (WindGusDir, WindGuSpeed), as well as humidity levels (Humidity9am, Humidity3pm), atmospheric pressure (Pressure9am, Pressure3pm), cloud cover (Cloud9am, Cloud3pm), and temperature readings at specific times (Temp9am, Temp3pm). Additionally, wind direction and speed measurements at specific times (WindDir9am, WindDir3pm, WindSpeed9am, WindSpeed3pm) are included. Crucially, the dataset also features indicators of precipitation such as RainToday (whether it rained on a given day) and RainTomorrow (prediction of rain the following day). These diverse features provide a rich foundation for analyzing weather patterns, understanding meteorological trends, and developing predictive models for rainfall occurrences, making it a valuable resource for meteorologists, climate scientists, and data analysts alike.

C. Classification

In our weather dataset analysis, classification involves predicting whether it will rain tomorrow based on the given meteorological data. Each row represents a day's weather observations, and the binary label indicates whether rain occurred the following day (RainTomorrow: 1 for 'Yes' and 0 for 'No'). The task is to accurately predict these binary labels for each day in the dataset. Suppose we are predicting the labels of N nodes in a test set, then the accuracy of the prediction is defined as:

$$\text{Accuracy} = N'/N \text{ in } [0, 1]$$

where (N') denotes the number of nodes for which the rain prediction for the next day is correctly predicted, and (N) is the total number of nodes in the test set. Achieving a high accuracy in this binary classification task is crucial for accurately forecasting rain occurrences, which has significant implications for various sectors such as agriculture, transportation, and disaster management.

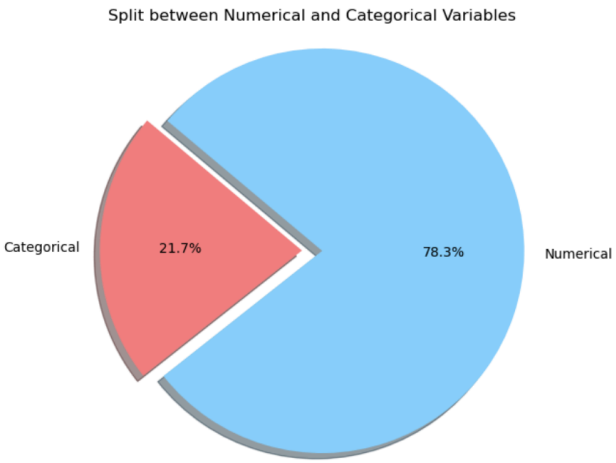
II. Pre-Processing

The preprocessing steps for the rain dataset are essential to ensure data quality and suitability for machine learning model training. Here's a detailed overview of the preprocessing steps: Handling Missing Values: The dataset is checked for missing values across all columns, and appropriate strategies such as imputation using median for numerical features and mode for categorical features are applied to fill missing data. Feature Engineering: The date column is converted to datetime format, and new features like 'Year,' 'Month,' 'Day,' and 'Season' are derived from it. Seasonal information is particularly valuable in weather-related predictions. Outlier Detection and Removal: Z-score-based outlier detection is performed on numerical features to identify and remove extreme values that could distort model training and predictions.

Encoded Categorical Variables:

Categorical columns such as 'Location,' 'WindGustDir,' 'WindDir9am,' 'WindDir3pm,' and 'Season' are encoded using Label Encoding to convert them

into numerical format for machine learning algorithms.



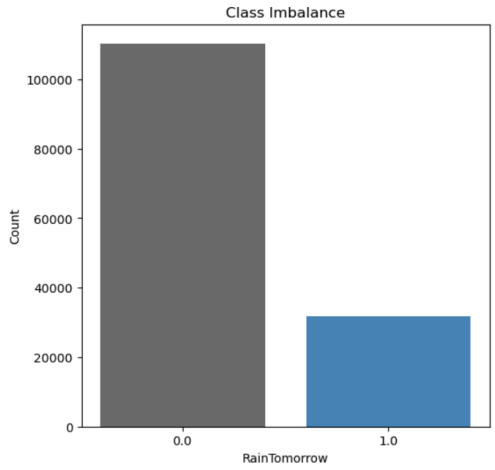
Feature Scaling:

Numerical columns are scaled using Min-Max scaling to bring all features to a common scale, preventing any single feature from dominating the model training process.

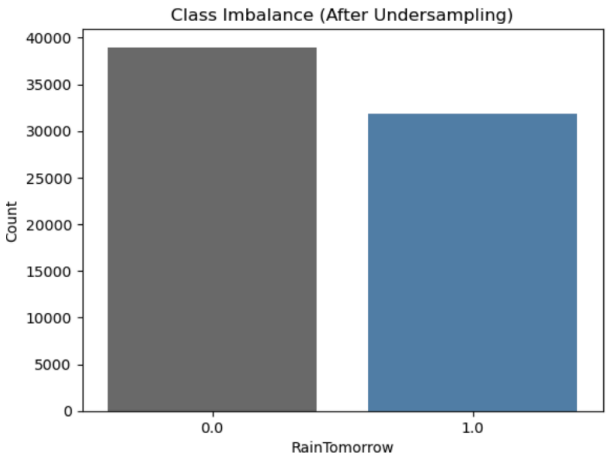
These preprocessing steps ensure that the rain dataset is cleaned, normalized, and ready for further analysis and predictive modeling tasks such as binary classification for rain prediction ('RainTomorrow'). The processed dataset retains essential information while mitigating data quality issues and ensuring model performance reliability.

III. Class Imbalance

Class imbalance refers to the disproportionate distribution of instances between the classes 'RainTomorrow' (indicating rain) and 'No rain tomorrow' (indicating no rain). This imbalance can significantly impact the performance of machine learning models trained on this dataset, particularly for binary classification tasks predicting rain occurrences. Originally, the class imbalance was as shown in the figure

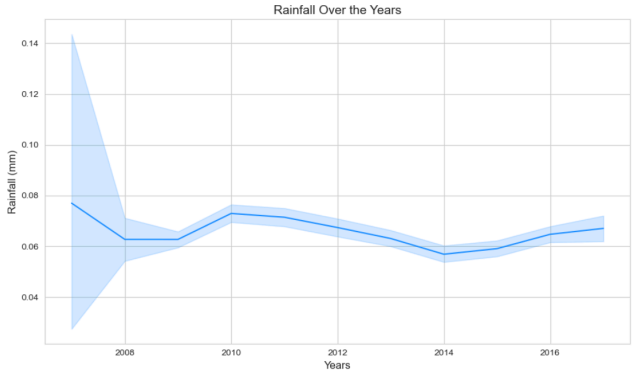


Consequently, we did undersampling to get rid of this class imbalance. Undersampling, a key preprocessing step in the weather dataset project, addressed class imbalance by reducing the instances of the majority class ('No rain tomorrow') to achieve a more balanced representation with the minority class ('RainTomorrow'). This technique ensured that machine learning models learned from both rainy and non-rainy conditions effectively during training, improving the model's fairness and accuracy in predicting weather outcomes. Visualizations such as count plots helped confirm the successful balance achieved post-undersampling, contributing to more reliable weather forecasts by mitigating biases towards the dominant class and enhancing the model's generalization capabilities.
Resultantly,

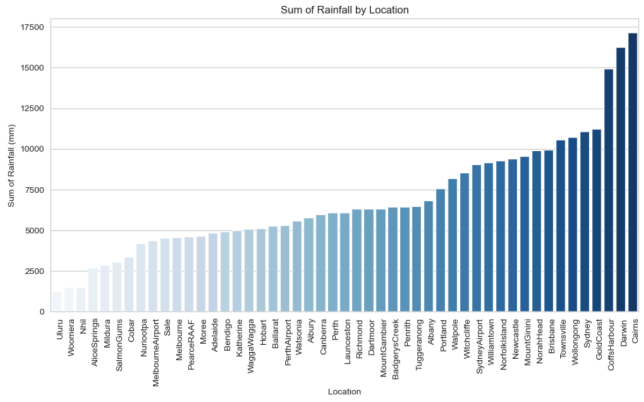


IV. Visualizations

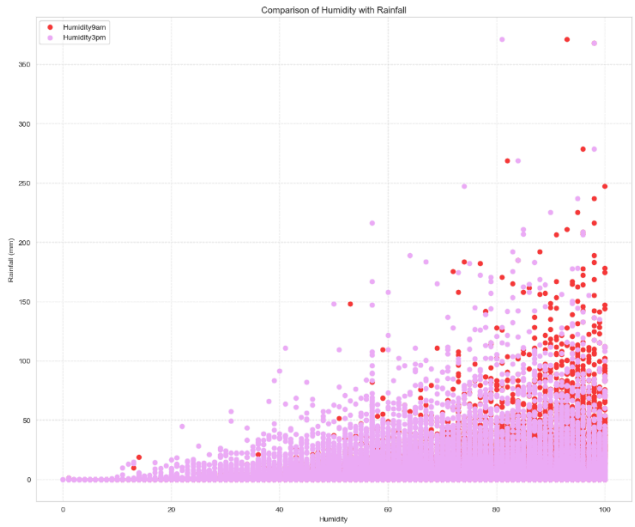
The visualizations in the weather dataset project played a crucial role in gaining insights into various aspects of weather data. For instance, the correlation heatmap highlighted multicollinearity among variables, aiding feature selection and model building. Time series plots illustrated trends in rainfall over the years, offering a historical perspective on weather patterns.



Monthly rainfall and sunshine visualizations provided seasonal variations, guiding seasonal forecasting and planning. Bar charts and histograms for categorical and numerical features revealed data distributions and class distributions, essential for preprocessing and modeling decisions.



Scatter plots depicting relationships between temperature, pressure, humidity, and rainfall helped understand their impact on weather conditions. Wind direction histograms showcased wind patterns, valuable for understanding regional climate characteristics. Relationship between humidity and rainfall.



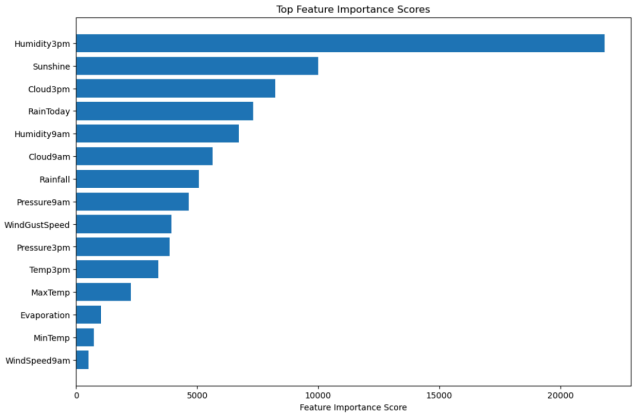
Relationship between humidity and rainfall.

Overall, these visualizations enriched the exploratory data analysis process, enhancing the understanding of weather variables and supporting informed decision-making in weather forecasting and analysis tasks.

V. Feature Extraction

In the feature extraction step of the project, several techniques were employed to identify and select the most relevant features for predicting the target variable, 'RainTomorrow.' Initially, the dataset was divided into features (X) and the target variable (y). Next, the SelectKBest method from scikit-learn was applied to perform feature selection based on their statistical significance. By using the f_classif scoring function and setting the number of desired features (k_best_features), the top features most relevant to predicting rain tomorrow were selected. The selected feature indices and names were then obtained to understand which features were chosen

based on their importance scores. Visualizing the feature importance scores using a horizontal bar plot provided a clear view of the top features contributing significantly to the prediction task. This process ensured that the model focused on the most impactful features, enhancing predictive performance while reducing computational complexity by excluding less relevant features.



Most useful features

VI. Model Training & Evaluation

Following, evaluation of multiple machine learning models was done for predicting the 'RainTomorrow' variable using a dataset that has undergone feature extraction and selection processes. The goal is to assess and compare the performance of various classification algorithms in predicting whether it will rain tomorrow.

The following machine learning models were trained, evaluated, and compared: Multi-Layer Perceptron (MLP) Classifier,Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, AdaBoost, Gaussian Naive Bayes (GaussianNB), Extra Trees, and CatBoost. Each model was trained on the selected features and evaluated using the following metrics: Accuracy, Precision, Recall, F1 Score, ROC-AUC, Cohen's Kappa

Additionally, confusion matrices and ROC-AUC curves were plotted to visualize the model performance and error analysis. Following is the comparison.

	Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC	Cohen's Kappa
0	MLP Classifier	0.790994	0.795748	0.712077	0.751591	0.871355	0.572156
1	Logistic Regression	0.779982	0.771978	0.715984	0.742928	0.858766	0.551103
2	Decision Tree	0.715643	0.677155	0.687277	0.682178	0.712745	0.424936
3	Random Forest	0.795972	0.786563	0.741804	0.763528	0.875223	0.584386
4	Gradient Boosting	0.788505	0.780732	0.728215	0.75356	0.866492	0.568723
5	AdaBoost	0.776889	0.769358	0.710549	0.738785	0.856928	0.544613
6	GaussianNB	0.75132	0.726319	0.705962	0.715996	0.818081	0.494904
7	Extra Trees	0.793634	0.783312	0.739935	0.761006	0.874282	0.579688
8	CatBoost	0.799894	0.791037	0.74656	0.768155	0.88163	0.592404

VII. Hyper-Parameter Tuning

Moreover, hyperparameter tuning and evaluation for various machine learning models using Python libraries such as Scikit-Learn, Matplotlib, Seaborn, and CatBoost was done. Let's break down the key steps and insights gained from this code.

Firstly, the code initializes and tunes several classification models: MLP Classifier, Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier (GBM), AdaBoost Classifier, Extra Trees Classifier, and CatBoost Classifier. Each model undergoes hyperparameter tuning using techniques such as RandomizedSearchCV or GridSearchCV to find the optimal set of hyperparameters for improved model performance.

After tuning, the code evaluates each model using metrics like accuracy, precision, recall, F1 score, ROC-AUC score, and Cohen's Kappa coefficient. These metrics provide a comprehensive understanding of how well each model performs on the test dataset, allowing for comparison and selection of the best-performing model.

The best hyperparameters found for each tuned machine learning model:

MLP Classifier:
The MLP Classifier was optimized using RandomizedSearchCV, and the best hyperparameters included {'learning_rate_init': 0.001, 'hidden_layer_sizes': (100,), 'alpha': 0.0001}

Logistic Regression: Logistic Regression's best hyperparameters, obtained through GridSearchCV, comprised{'solver': 'saga', 'penalty': 'l2', 'max_iter': 2000, 'C': 1.0}

Decision Tree Classifier: Hyperparameter tuning with RandomizedSearchCV for Decision Tree Classifier yielded {'criterion': 'gini', 'max_depth': 7, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 2}

Random Forest Classifier: Random Forest Classifier's best hyperparameters were{'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': None, 'criterion': 'gini'} obtained via RandomizedSearchCV.

Gradient Boosting Classifier (GBM): GBM's optimal hyperparameters, found through RandomizedSearchCV, included{'n_estimators': 100, 'max_depth': 5, 'learning_rate': 0.15

AdaBoost Classifier: AdaBoost Classifier's best hyperparameters obtained from RandomizedSearchCV {'n_estimators': 100, 'learning_rate': 1.0, 'algorithm': 'SAMME.R'}

Extra Trees Classifier: For Extra Trees Classifier, the best hyperparameters included {'n_estimators': 100, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': None, 'bootstrap': False} determined using RandomizedSearchCV.

CatBoost Classifier: CatBoost Classifier, being optimized inherently, was fine-tuned with {'border_count': 73, 'depth': 10, 'iterations': 150, 'l2_leaf_reg': 5, 'learning_rate': 0.15}

These hyperparameter configurations represent the best settings found during the hyperparameter tuning process for each respective model,contributing to improved model performance and predictive accuracy across the classification tasks.

Summary of hyper-parameter tuned models,

Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC	Cohen's Kappa
MLP Classifier	0.790994	0.795748	0.712077	0.751591	0.871355	0.572156
Logistic Regression	0.780057	0.772119	0.715984	0.742993	0.858767	0.551249
Decision Tree Classifier	0.767914	0.767619	0.684559	0.723714	0.843516	0.524762
Random Forest Classifier	0.795143	0.783987	0.743503	0.763208	0.877746	0.582919
Gradient Boosting Classifier	0.797481	0.789826	0.741125	0.764701	0.875897	0.587261
AdaBoost Classifier	0.783602	0.780379	0.713436	0.745408	0.861664	0.557896
Extra Trees Classifier	0.793634	0.783312	0.739935	0.761006	0.874282	0.579688
CatBoost Classifier	0.797933	0.788489	0.744692	0.765965	0.878544	0.588438

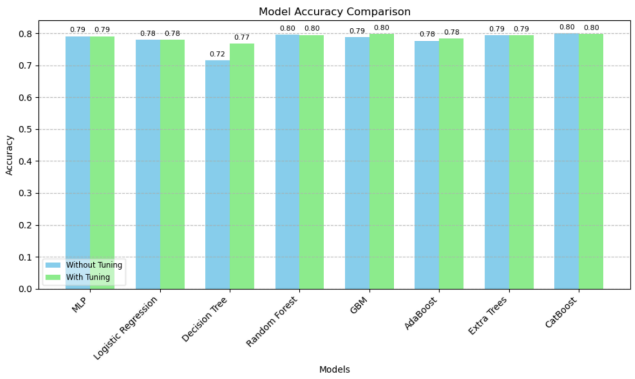
VIII. Comparison with and without Hyper-parameter Tuning

A bar chart is presented to compare the accuracy scores of different machine learning models with and without hyperparameter tuning. The models considered include MLP, Logistic Regression, Decision Tree, Random Forest, GBM (Gradient Boosting), AdaBoost, Extra Trees, and CatBoost classifiers.

The code defines two lists: “accuracy_scores_without_tuning” and “accuracy_scores_with_tuning”, which store the accuracy scores of each model without and with hyperparameter tuning, respectively. These scores are then used to create a bar chart using Matplotlib.

The bar chart displays side-by-side bars for each model, comparing the accuracy scores before and after hyperparameter

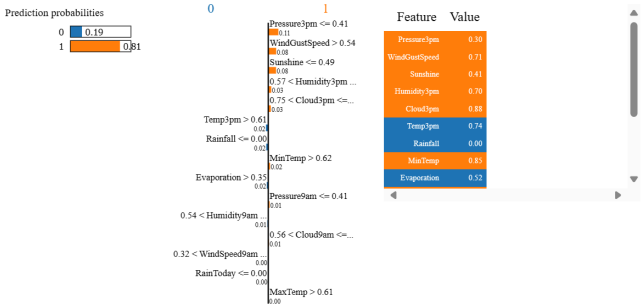
tuning. The x-axis represents the different models, while the y-axis represents the accuracy values. Grid lines are added for clarity, and the bars are labeled with their respective accuracy values for easy comparison.



The chart is annotated with model names, accuracy values, and labeled axes to provide a clear visualization of how hyperparameter tuning has impacted the accuracy of each machine learning model. Overall, it provides a visual comparison of model performance improvements due to hyperparameter optimization.

XI. Explainable AI (LIME)

Used LIME (Local Interpretable Model-agnostic Explanations) for explaining individual predictions of a machine learning model trained on tabular data. Initially, the feature_names list is defined to represent the column names based on the dataset features, which is crucial for interpreting the explanations generated by LIME. The LIME explainer (explainer_lime) is then initialized incorporating the training data (X_train), corresponding training labels (y_train), and feature names list (feature_names). A specific instance from the test set is selected for explanation through instance_idx, and the LIME explainer is used to generate explanations for this instance using the explain_instance method. Finally, the LIME explanation is displayed in a Jupyter notebook environment using exp_lime.show_in_notebook(), providing insights into the crucial features and their contributions to the model's prediction for the chosen instance, aiding in model interpretability and analysis.



X. Conclusion

In conclusion, the project demonstrated the effectiveness of machine learning in rainfall prediction, highlighting the importance of data quality, feature selection, model optimization, and interpretability in developing accurate and reliable predictive models. The insights gained contribute not only to meteorological research but also to practical applications in sectors reliant on weather forecasting for planning and decision-making. Future work may include exploring ensemble methods, incorporating real-time data for dynamic predictions, and expanding the scope to regional or global weather modeling for broader impact and applications.