رواد مصر الرقمية

Team SkyCast

RAIN PREDICT
PROJECT

Supervised by **Eng. Heba**

Team SkyCast

OUR TEAM MEMBERS

1. Hassan Abdelrazek (Team Leader)
2. Mahmoud Ebrahim
3. Mohamed Elseragy
4. Abdulrhman Hosny
5. Ahmed Fouad
6. Wageeh Abdelhameed

Agenda

01 Introduction

02 Our Team

03 Project Workflow

04 Dataset Overview

05 Initial Data Exploration (EDA)

06 Data Cleaning & Preprocessing

07 Feature Engineering & Selection

08 Modeling Strategy &Evaluation

09 Deployment

10 Future Work

Smart Rain Forecasting Step-by-Step Workflow

01 Problem Understanding

- Define the goal: Predict rain using weather data.
- Understand the business value and real-world impact

02 Data Collection

- Get the weatherAUS.csv dataset.
- Explore data source, structure, and formats.

Initial Data Exploration (EDA) 03

- Identify patterns, distributions, and relationships.
- Visualize rain patterns, temperature, humidity, etc.

04 Data Cleaning

- Handle missing values.
- Detect and treat outliers.
- Check for data redundancy.

05 Preprocessing

- Scale numerical features.
- Encode categorical variables.
- Transform skewed distributions

Feature Engineering& Selection 06

- Create new useful features.
- Perform statistical tests and correlation analysis.
- Selectmost relevant features for modeling.

Modeling 07

- Train models: XGBoost, Random Forest, Decision Tree.
- Use pipeline with preprocessing steps.
- Apply cross-validation and parameter tuning.

Evaluation 08

- Use proper metrics: F1, Recall, ROC-AUC.
- Handle class imbalance, overfitting issues

Result Interpretation 09

- Document the troubleshooting steps, solutions, and any new knowledge gained during the process

Model Deployment 10

built a simple interactive dashboard using Streamlit

I will cover some steps

let's Go >>

Problem Understanding

Why Predict Rain?

Helps people plan their day (travel, clothing, etc.)

Supports agriculture and water resource management

Useful in weather alert systems

Data Collection

Dataset Overview

01

Source & Context:

The data was collected from weather stations across Australia, providing detailed daily weather observations.

02

Volume & Variety:

It includes over 145,000 records with a mix of numerical features (like temperature and humidity) and categorical ones (like wind direction and location).

03

Why It Matters for Us:

The dataset's size and diversity made it suitable for building strong predictive models — but also introduced real-world complexity that we had to handle early.

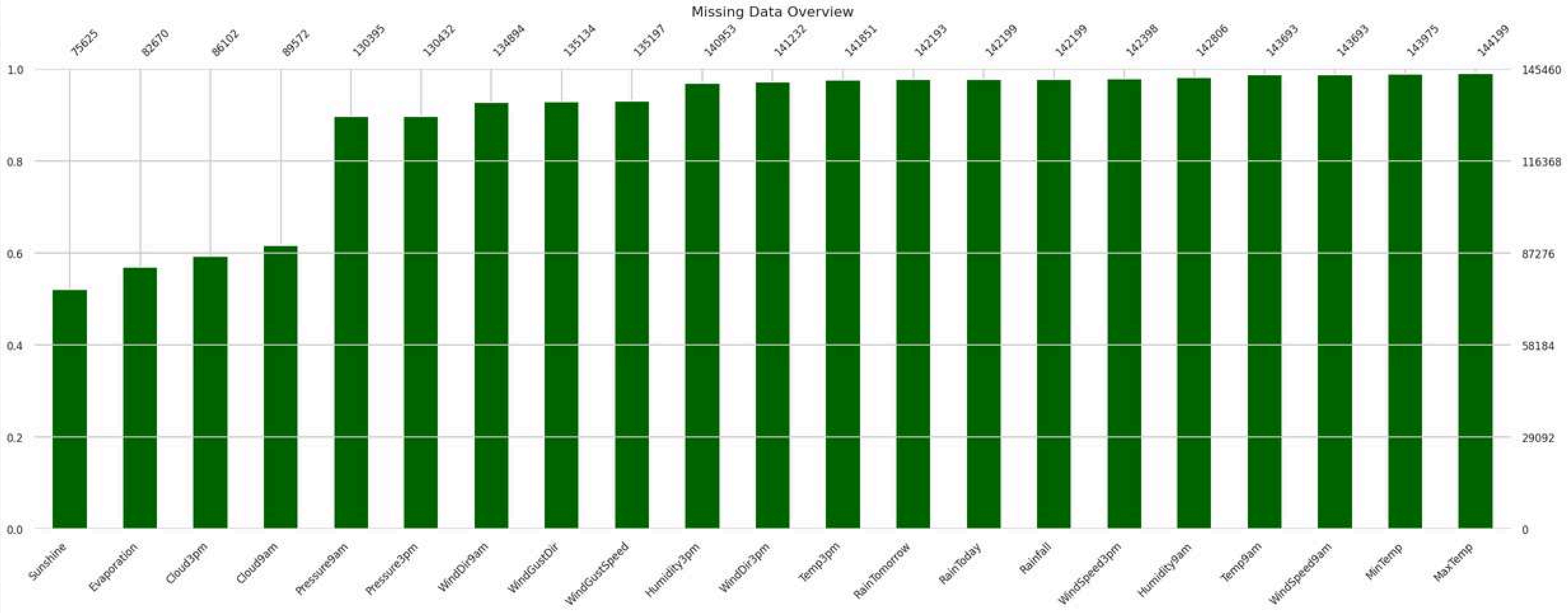
Initial Data Exploration

Initial Data Inspection

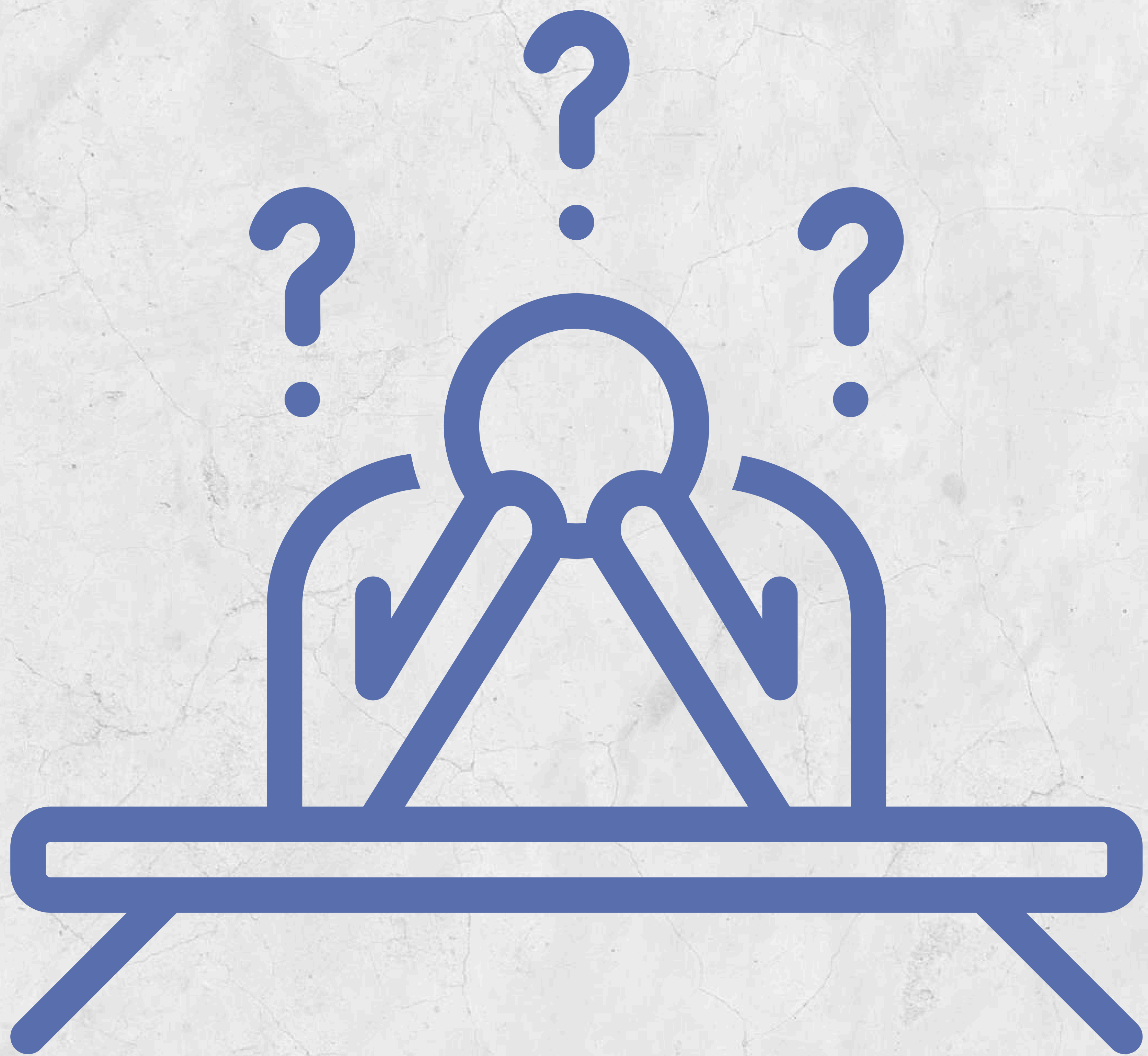
What did we see when we first opened the dataset?

Key Findings:

- 145,460 rows × 24 columns
- Mixture of numerical, categorical, and date features
- missing values in columns reached ~40% in some columns
- outliers in Rainfall
- Date Column is object dtype
- Total Duplicates: 0



DATA CHALLENGES



DATA CHALLENGES ROADMAP

DATA QUALITY ISSUES

Missing Values
Skewed Data

DATA IMBALANCE

RainTomorrow
No Rain
Accuracy
Evaluation

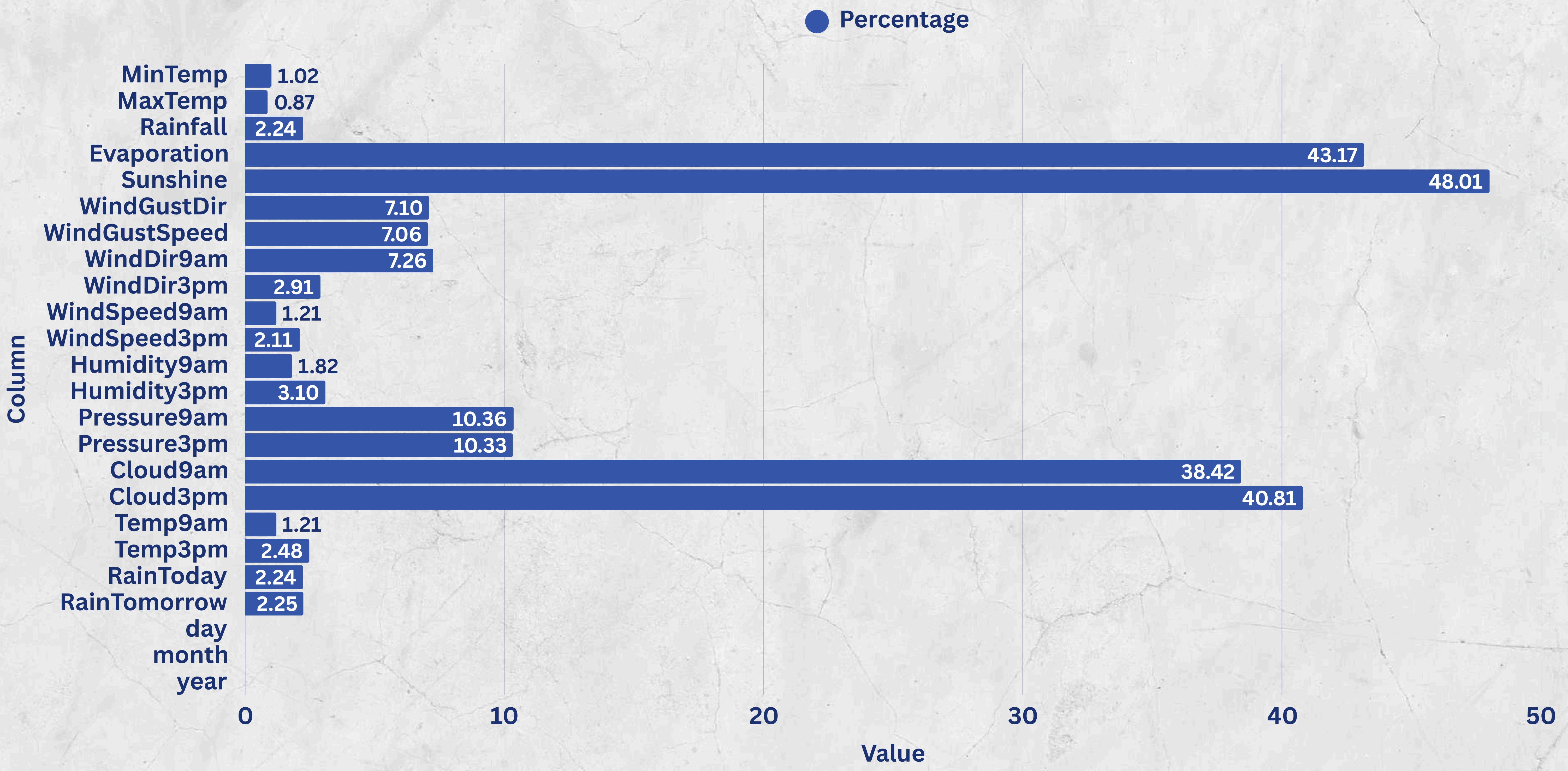
ENCODEIN FOR CAT

binary
label
enable cat
(xGBoost)

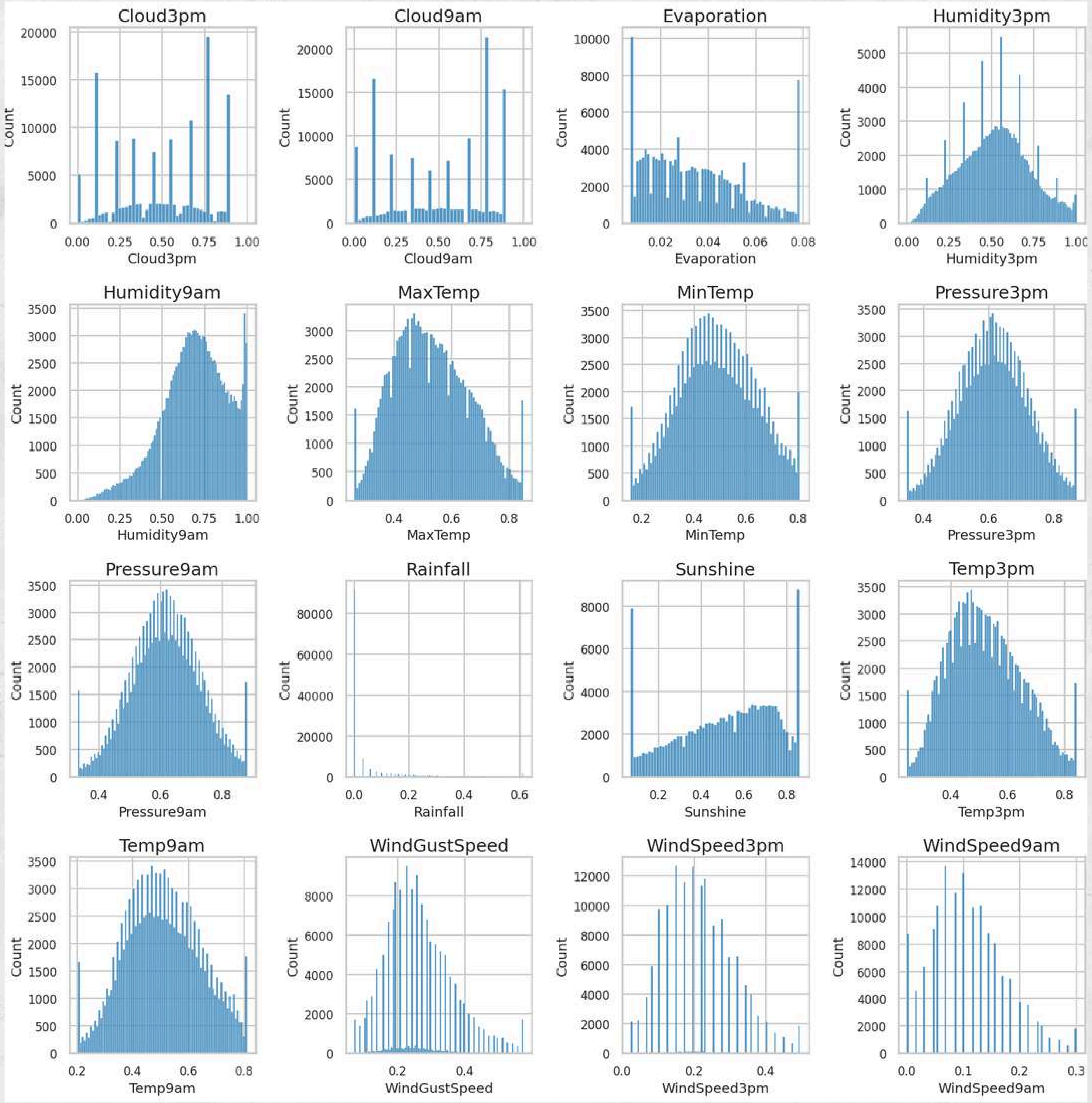
MODEL SELECTION

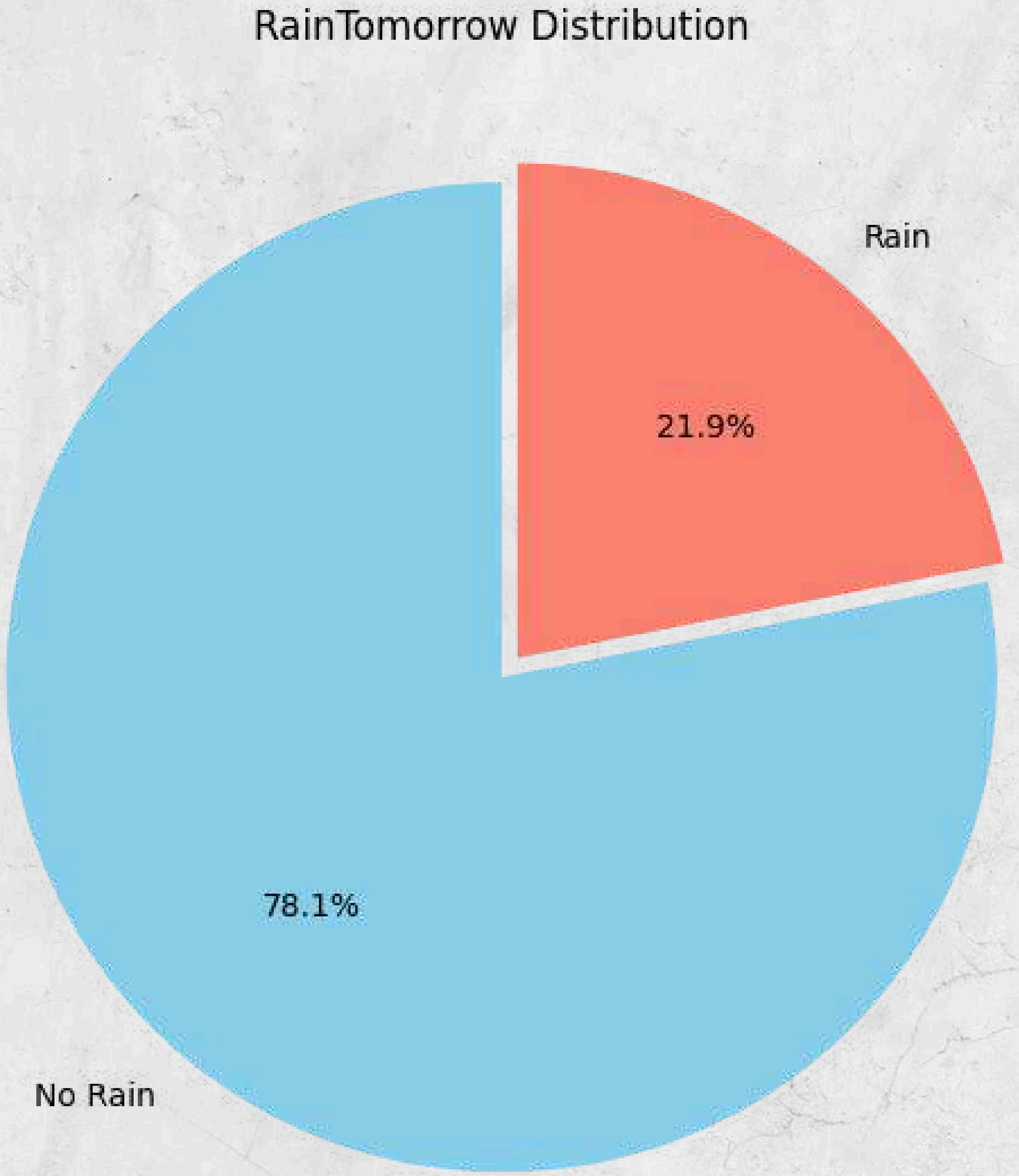
XGBoost
Random forest
Decision Tree

Missing Values



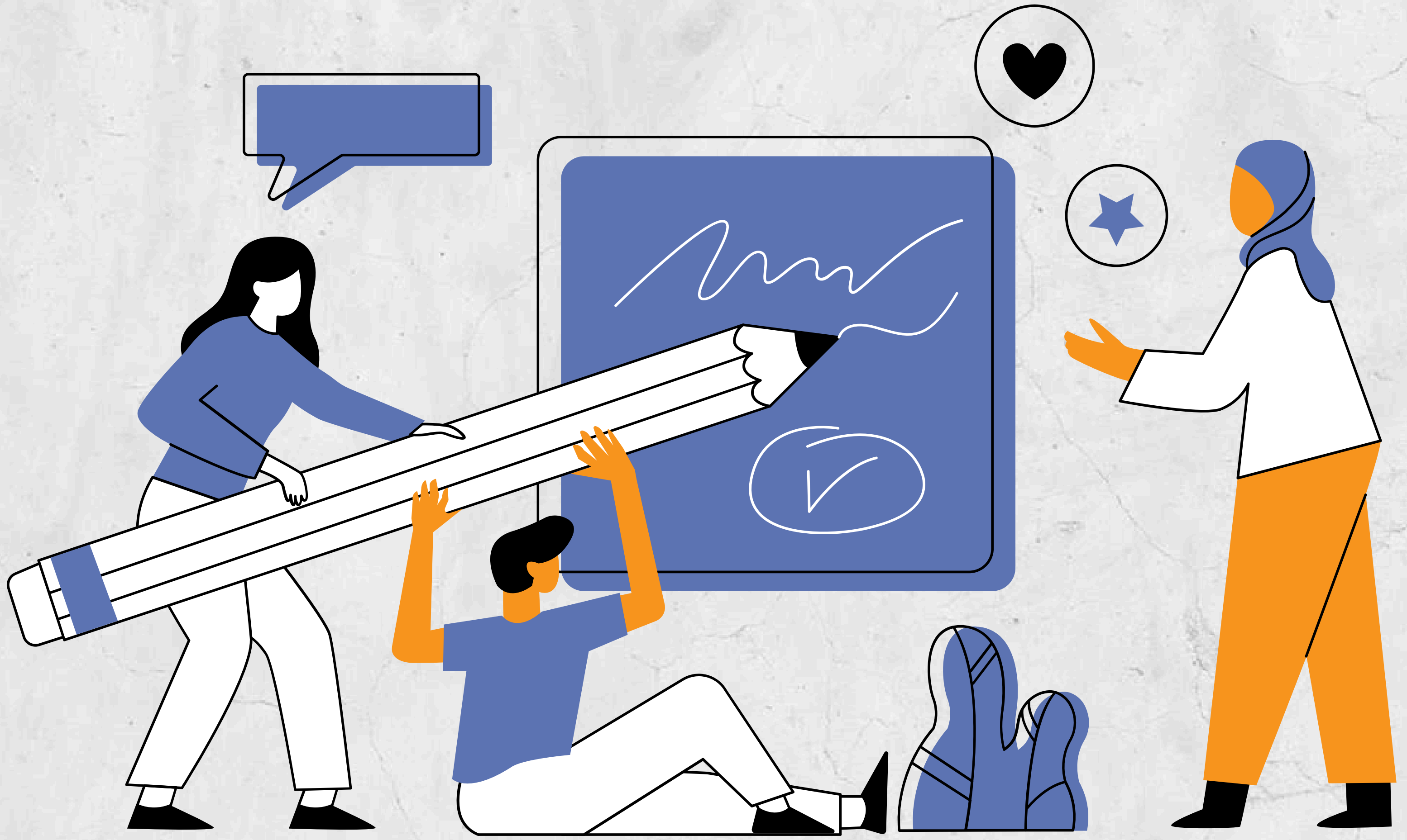
Features Distribution





RainTomorrow Distribution

- Majority of days have no rain
- Data is imbalanced
- Needs to be considered in model training



Data Cleaning

Data Cleaning

How did we handle the data quality problems?
Handling Missing Values

Problem:

Many columns had missing values, especially in wind & humidity.

What We Did:

For numerical columns: Used KNN Imputer to fill based on nearest neighbors

For categorical columns: Filled using most frequent (mode).

Result:

Missing values filled with relevant, context-aware estimates.

No major loss of information or need to drop rows.

Handling Outliers

Problem:

Some numeric features had extreme outliers

What We Did:

Winsorize Method For Handling Outliers

Result:

- Keeps the sample size unchanged, unlike methods like trimming that remove values
- Reduced noise in training data.

Data cleaned… but the real work begins

Feature Engineering & Selection

Date Column Processing

Convert the Date column into datetime format and extract temporal features (day, month, year).

01 Revealed seasonal patterns (e.g., higher rain likelihood in Winter/Autumn).

02 Improved model ability to capture temporal dependencies.

03 Enable time-based analysis (e.g., seasonal trends).

Numerical Feature Engineering

- TempDiff: Captures daily temperature swings — smaller swings often indicate rain
- WindSpeedAvg, HumidityDiff, PressureDiff: Reflect intraday changes; pressure drops signal rain
- RainToday: Binary flag — rain events tend to cluster

Impact

- Higher HumidityDiff & lower PressureDiff associated with rain
- WindGustDiff ranked among top 5 predictive features
- Capturing dynamic weather interactions improved model accuracy

Seasonal Feature Creation

Purpose

Categorize months into seasons to capture cyclical weather patterns.

Benefit

- Identified Winter/Autumn as high-rain seasons
- Added domain knowledge to the model.

Preprocessing

Categorical Feature Encoding

To match each model's nature, we applied encoding techniques accordingly:

- Random Forest → Binary Encoding
- XGBoost → Handled internally (no manual encoding needed)
- Decision Tree → Label Encoding

This ensured optimal compatibility and preserved model performance

Statistical Tests

Chi-Square Test for categorical features

ANOVA Test for numerical features

To identify which features have a meaningful relationship with the target variable (RainTomorrow), and to guide our feature selection process.

Statistical Tests

Chi-Square

- Tested feature association with RainTomorrow
- Significant: **Location, RainToday, WindDir3pm, WindDir9am, WindGustDir**
- Not significant: Season

ANOVA

- Tested mean difference by rain outcome
- Significant: **MaxTemp**, **Rainfall**, **WindGustSpeed**, **Sunshine**, **Pressure3pm**

Features Selection

Features Selection

Methods Used

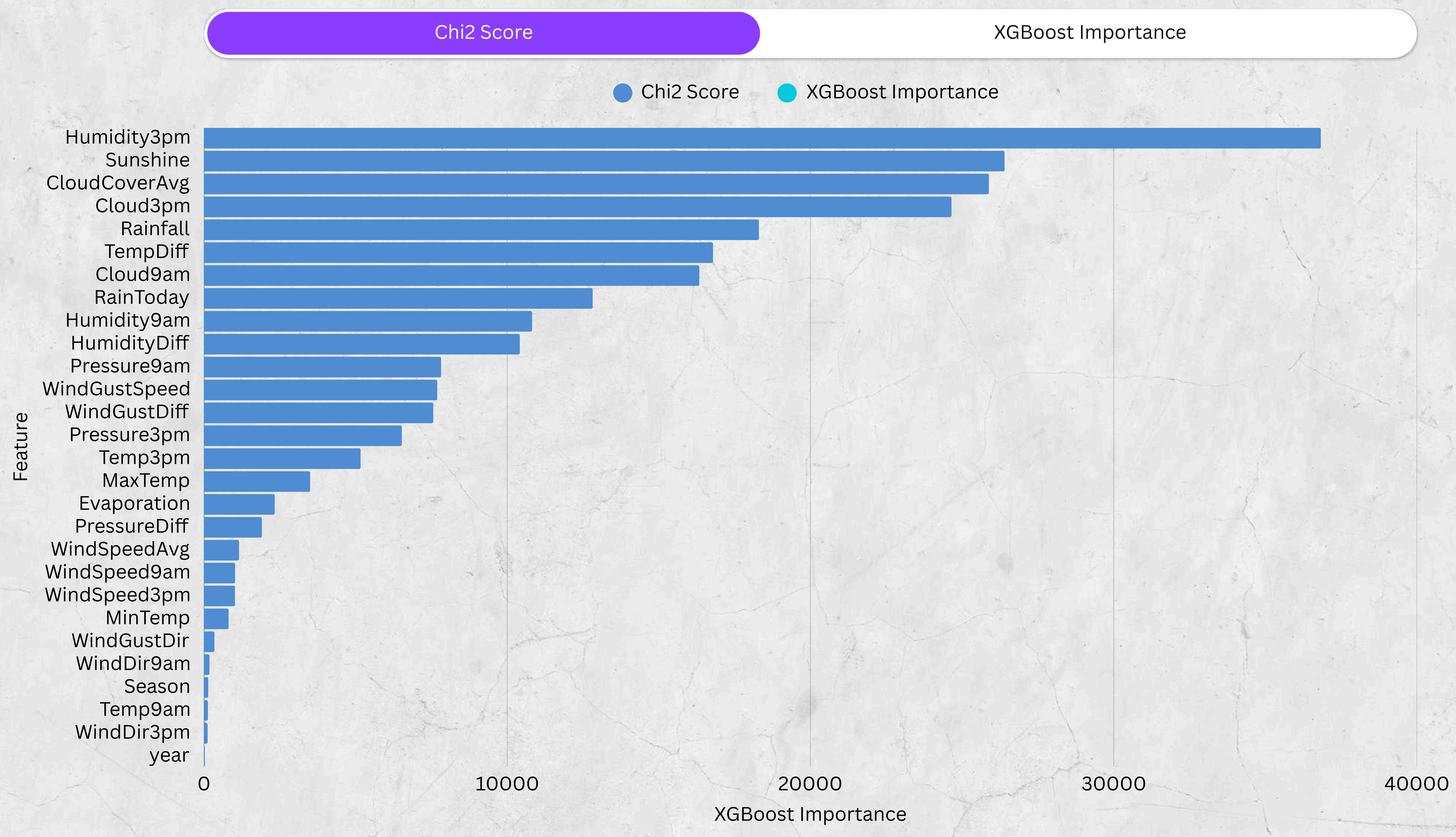
- Chi-Square Test (SelectKBest)
- XGBoost Feature Importance

Goal:

- Understand which features have the strongest relationship with the target (RainTomorrow)
- Improve interpretability and prepare for model training

Note:

- No features were dropped
- XGBoost handles low-importance features internally



Chi2 Score
XGBoost Importance
Chi2 Score
XGBoost Importance
Humidity3pm
Sunshine
CloudCoverAvg
Cloud3pm
Rainfall
TempDiff
Cloud9am
RainToday
Humidity9am
HumidityDiff
Pressure9am
WindGustSpeed
WindGustDiff
Pressure3pm
Temp3pm
MaxTemp
Evaporation
PressureDiff
WindSpeedAvg
WindSpeed9am
WindSpeed3pm
MinTemp
WindGustDir
WindDir9am
Season
Temp9am
WindDir3pm
year
Feature
0
10000
20000
30000
40000
XGBoost Importance

Clean & Ready!

The dataset is now fully cleaned and saved — ready for modeling

Model Selection

- Decision Tree
- Random Forest
- XGBoost

Why We Chose These Models?

- **High number of missing values**
- **Class imbalance in the target variable**
- **Presence of outliers**

Decision Tree Classifier

Used as a initial model to quickly test performance and feature influence

Random Forest Classifier

Designed to generalize well and improve stability over Decision Tree

XGBoost Classifier

Chosen as the main production model due to highest performance in tests

Each model we chose tackled a real problem in our data — our strategy wasn't random, it was data-driven."

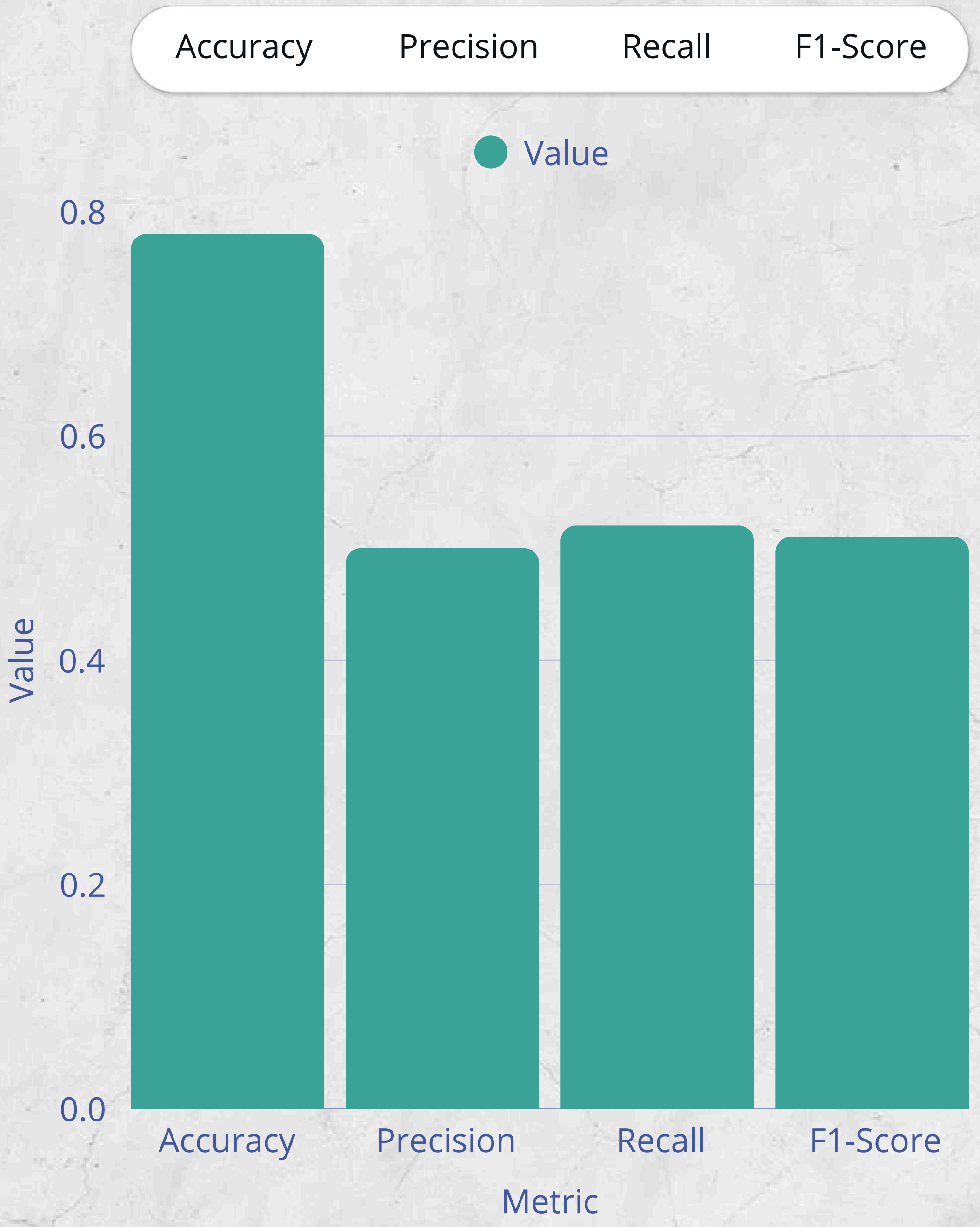
Decision Tree:Model Training & Evaluation

Before Tuning

- Used default Decision Tree
- No parameter tuning
- Weak performance on rain class
- Overfitting likely on training data

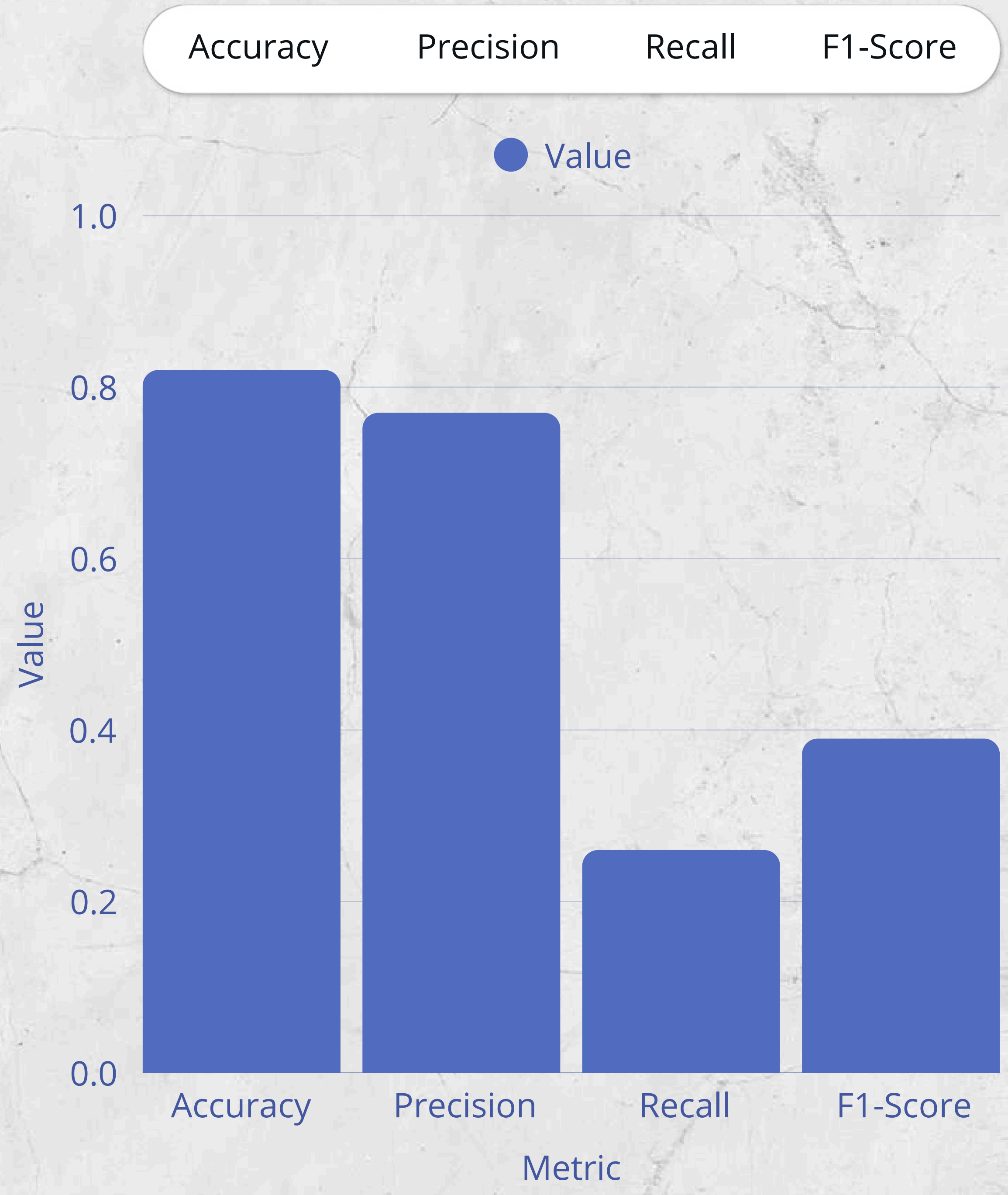
After Tuning

- Used GridSearchCV for best params
- Better overall accuracy
- Precision improved
- But recall dropped for rain cases



Before Tuning

After Tuning

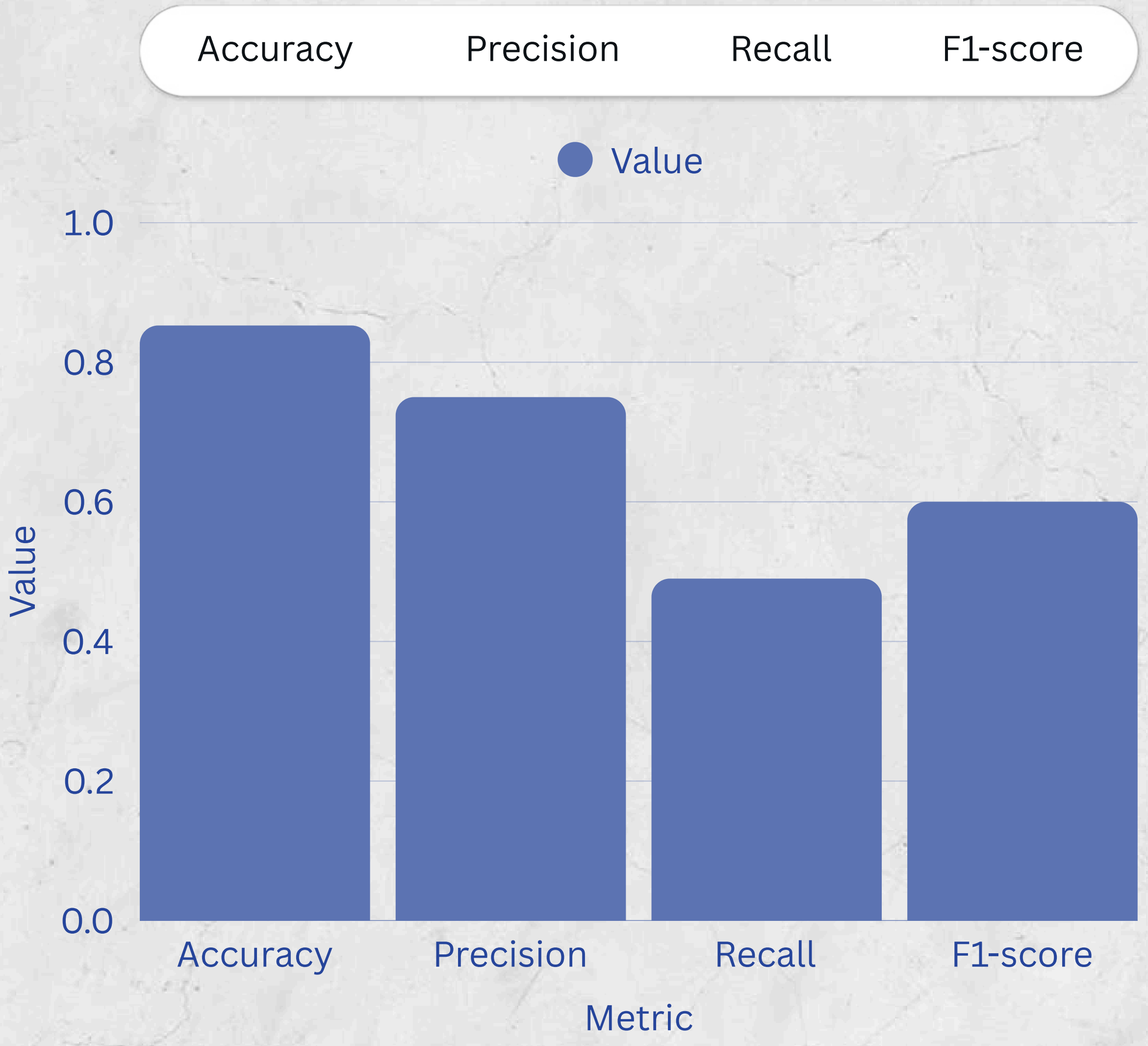


Random Forest :Model Training & Evaluation

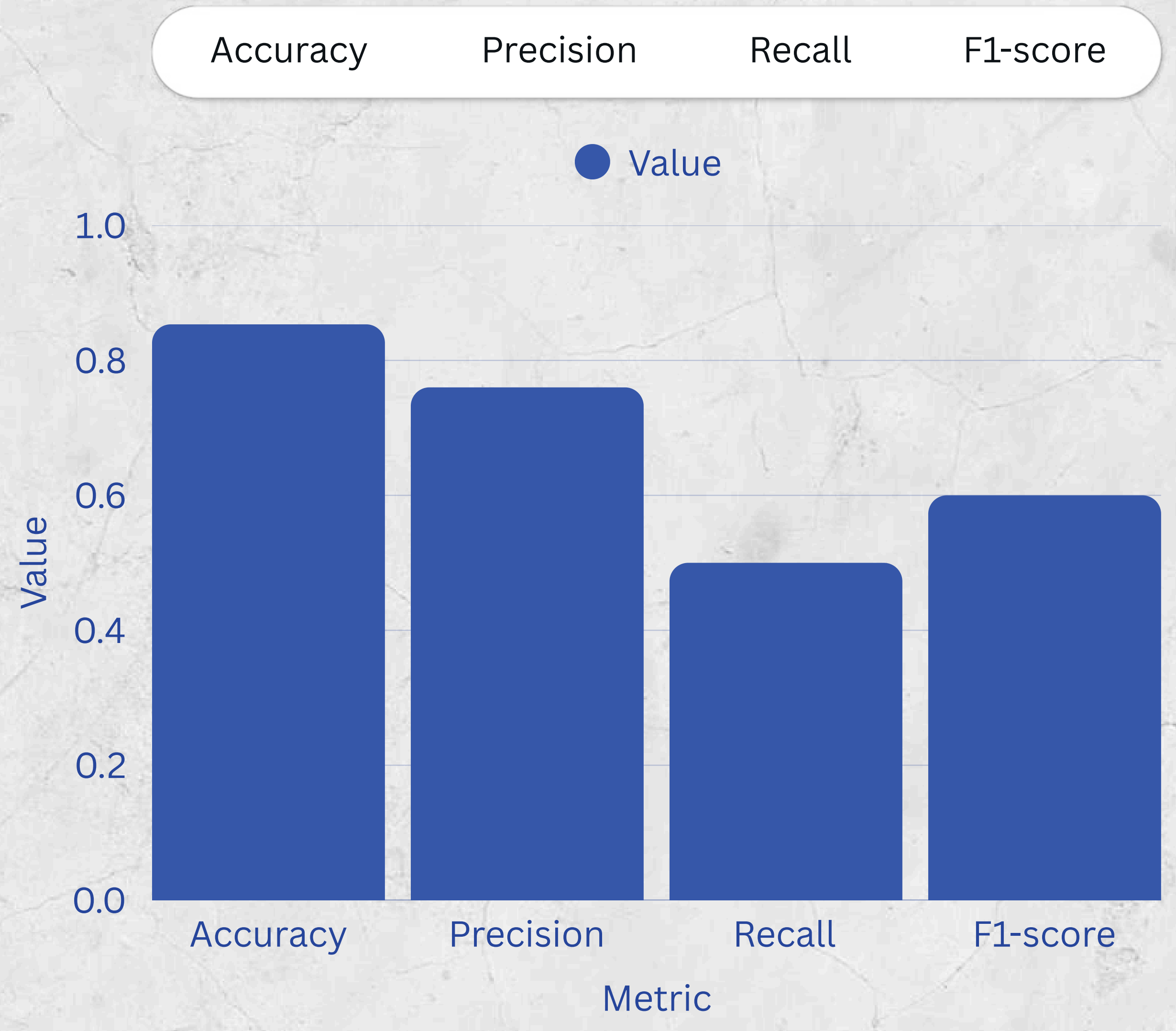
Before Tuning
- Used default Random Forest settings.

After Tuning
- Tuned with Bayesian Optimization.
- Best params: max_depth=30, n_estimators=200, etc.
- Applied SMOTE to fix class imbalance.
- Improved recall for "Rain = Yes".
- Training took longer, but performance more balanced.

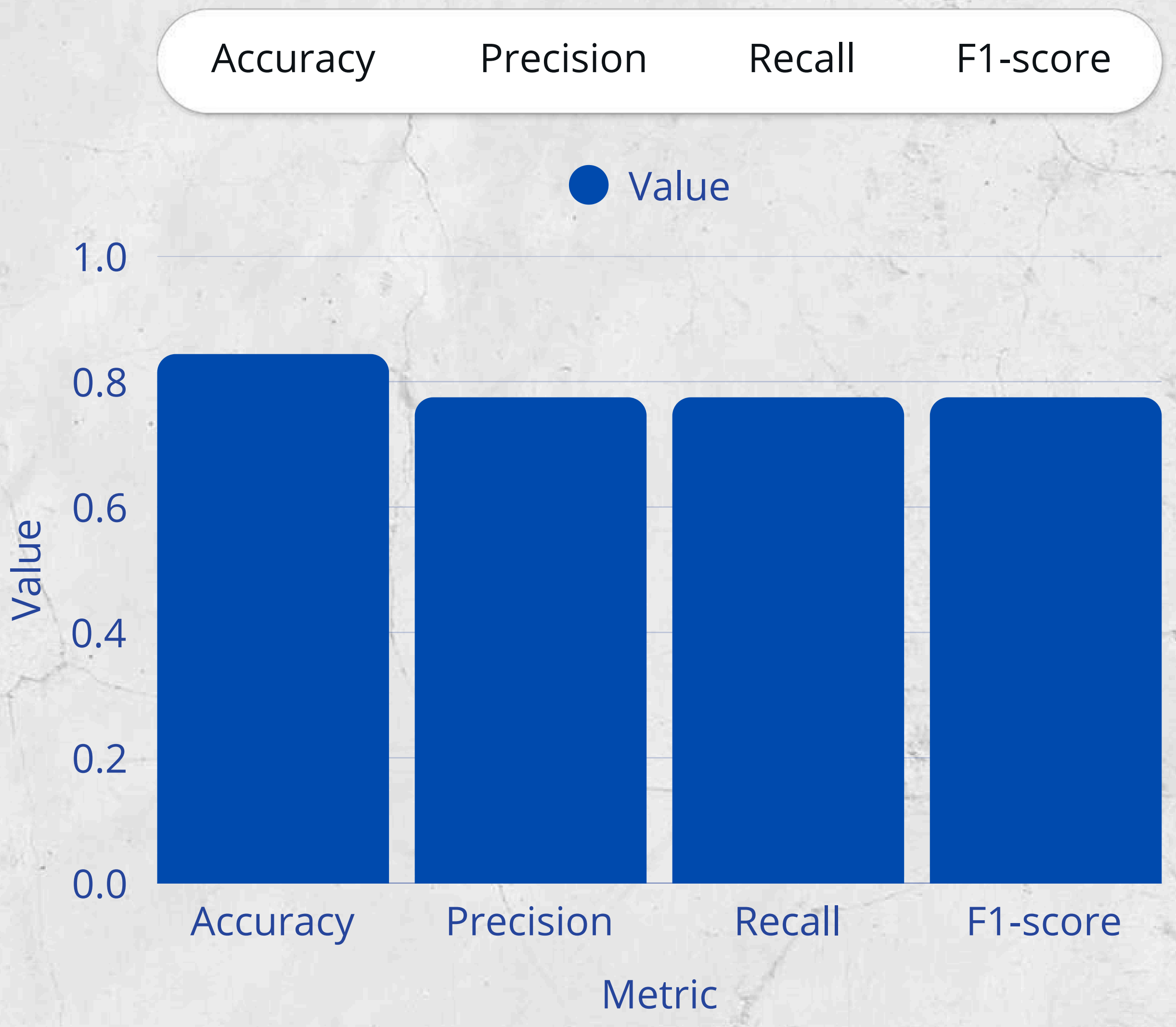
Before Tuning



After Tuning



After SMOTE



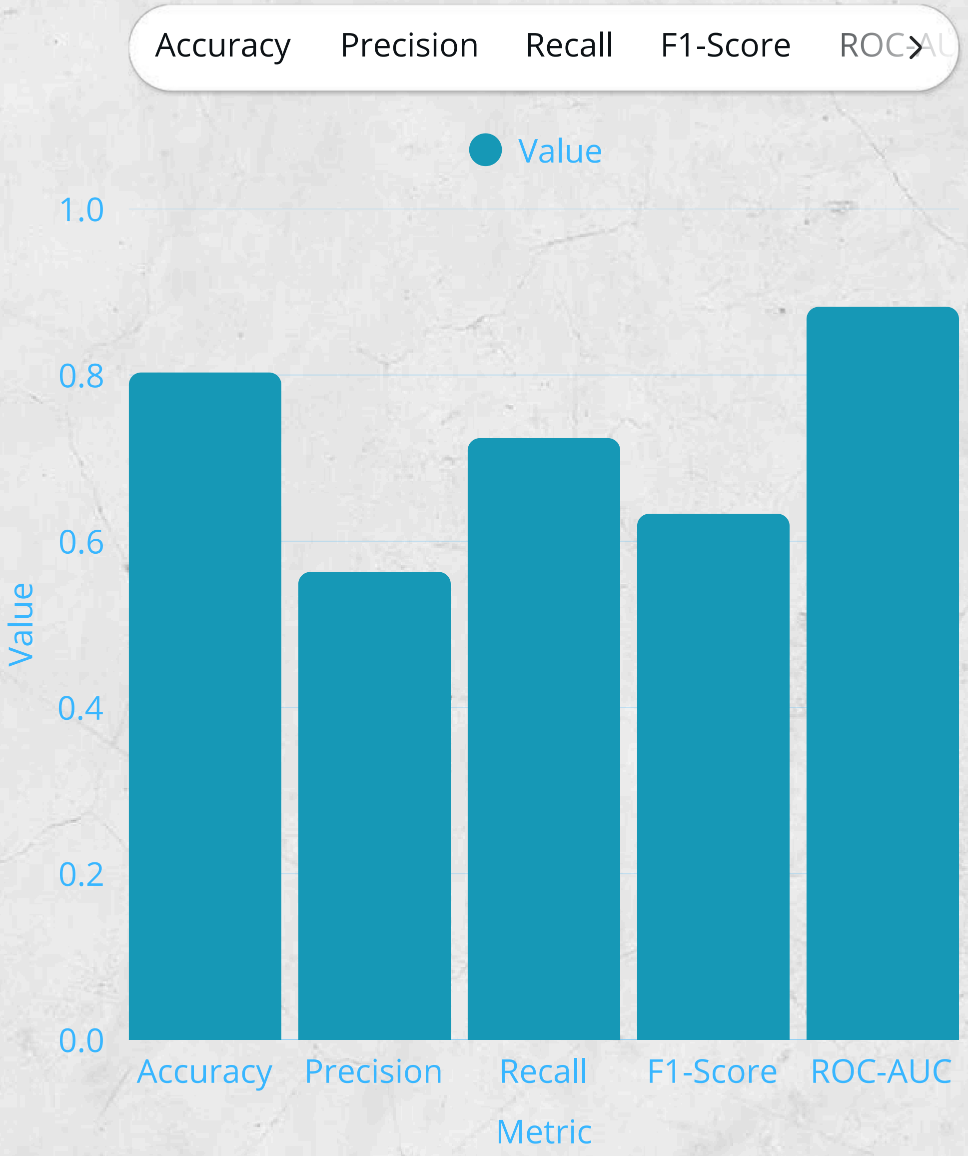
XGBoost:Model Training & Evaluation

Before Tuning

- Used default XGBoost settings
- No handling for class imbalance
- Fast training, but not optimal
- Weak on minority class (missed rain cases)
- Model was overconfident, needed regularization

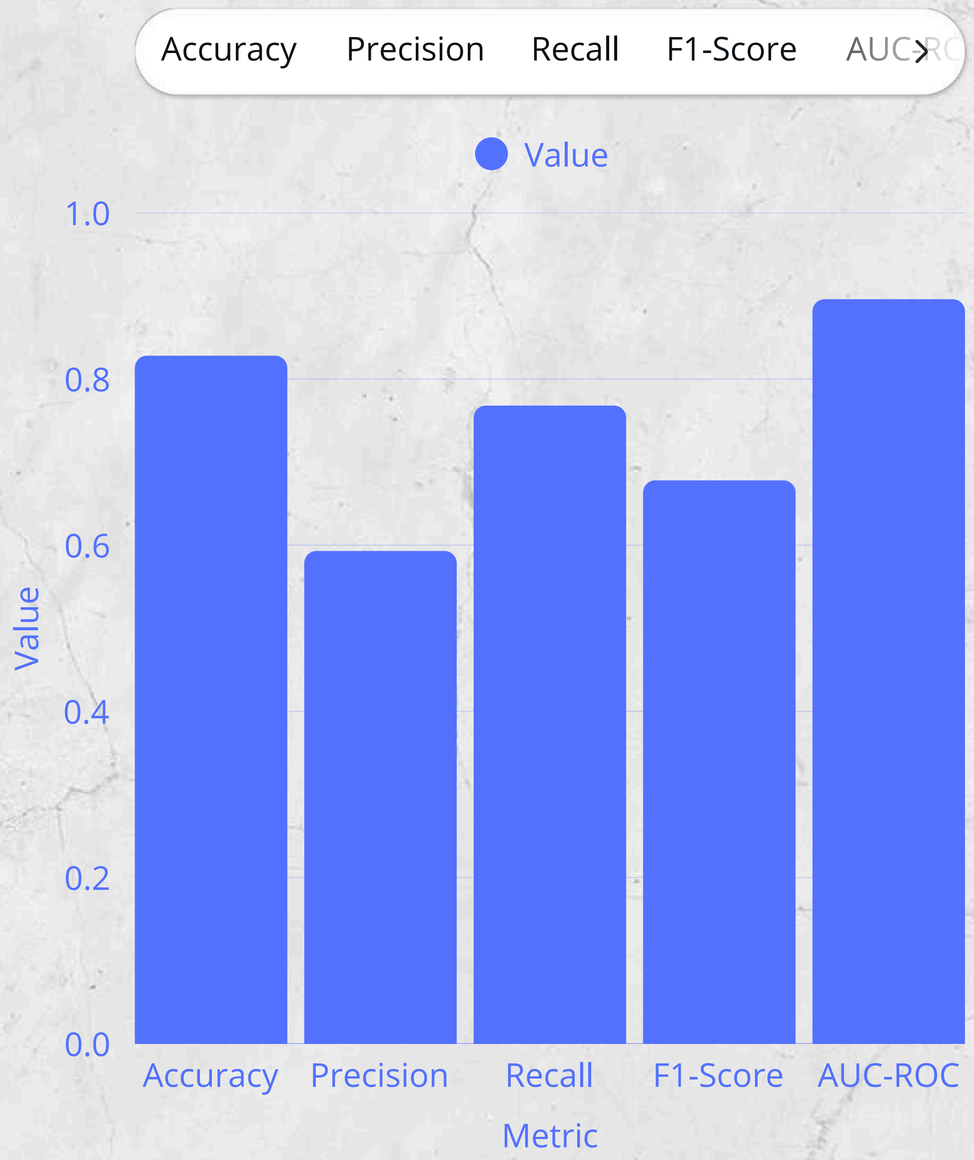
After Tuning

- Balanced classes with scale_pos_weight
- Reduced learning_rate for smoother learning
- Added regularization (gamma, lambda, alpha)
- Used early stopping for better generalization
- Tuned threshold to improve recall (detect more rain)

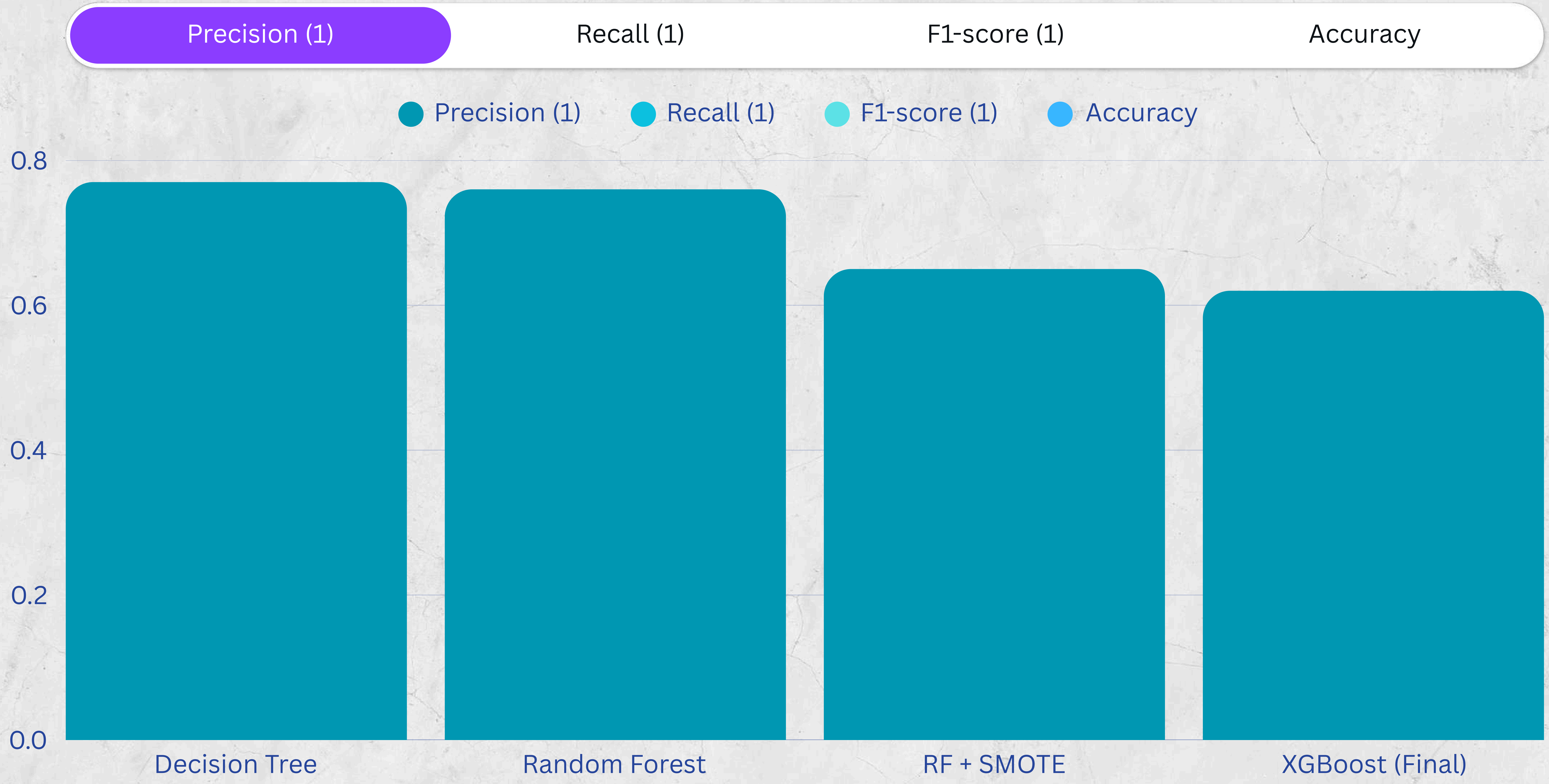


Before Tuning

After Tuning



Which Model Predicts Rain Best?



XGBoost: Our Final Model

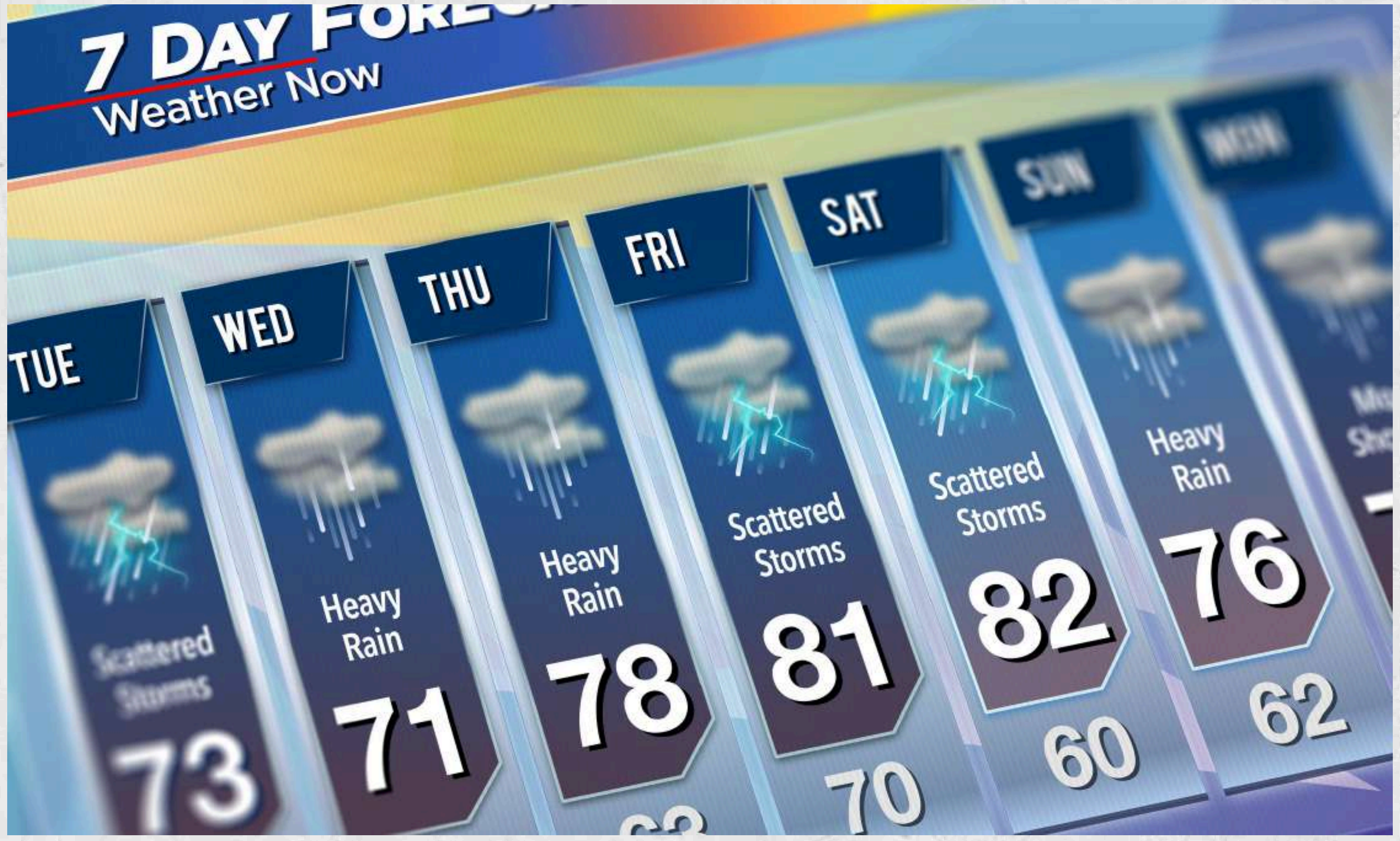
Why We Chose It:

Dataset Challenge	**XGBoost Solution**
Missing data	Handled without manual imputation
Outliers	Less sensitive compared to linear models
Class imbalance	Managed with weight tuning
Feature complexity	Learns nonlinear interactions well

Model Deployment

🌟 [See the Project in Action]

Future Work



01 Use real-time weather data

02 Improve model accuracy

03 Build a mobile app

05 Work on long-term prediction

Access the Project

Includes full exploratory data analysis (EDA), visualizations, and model evaluation.

[Kaggle]

Our GitHub Repo

Collaborative work with clean code, data preprocessing, and model training.

THANK YOU

Any Questions or Feedback?