**Introduction**

**ABSTRACT:**

Rainfall prediction plays a vital role in agricultural planning, water resource management, and disaster preparedness. Inaccurate or delayed forecasts can lead to devastating consequences, particularly in regions highly dependent on seasonal rainfall. Traditional methods often struggle with the complexity and variability of climatic data. In recent years, Machine Learning (ML) techniques have emerged as powerful tools for building reliable and data-driven predictive systems. This project aims to predict whether it will rain the next day using meteorological features such as temperature, humidity, pressure, and wind speed from the "weatherAUS" dataset. The model was developed and evaluated using various supervised learning algorithms including K-Nearest Neighbors (KNN), Logistic Regression, Decision Trees, and Random Forests. Techniques such as data normalization, handling missing values, and class imbalance resolution using SMOTE were also applied to improve performance. Among the tested models, the Random Forest algorithm showed the highest accuracy and robustness in predicting rainfall. This study demonstrates the potential of ML in enhancing weather forecasting systems and supporting timely decision-making for farmers and climate-sensitive sectors.

**DESCRIPTION**

Rainfall prediction is a vital task in sectors such as agriculture, water resource management, and disaster prevention. Accurate and timely predictions can help farmers make critical decisions regarding crop cycles, irrigation scheduling, and harvesting, while also supporting governments and agencies in preparing for potential natural disasters like floods or droughts. However, predicting rainfall is challenging due to the complexity and variability of weather conditions.

Traditional forecasting models often rely on physical simulations and statistical methods, which may not adequately capture the nonlinear relationships between meteorological variables. In contrast, Machine Learning (ML) offers a data-driven approach that can learn from historical weather data to uncover patterns and make predictions with improved accuracy.

This project focuses on building an ML-based system to predict whether it will rain tomorrow (Yes/No) based on a variety of weather-related features such as temperature, humidity, wind speed, pressure, and rainfall measurements. The model uses preprocessing techniques like handling missing values, normalizing the features, and removing outliers to prepare the data for training.

Several supervised machine learning algorithms are employed and compared, including K-Nearest Neighbors (KNN), Logistic Regression, Decision Tree, and Random Forest. The dataset used is the “weatherAUS” dataset provided by the Australian Bureau of Meteorology, containing daily weather observations across multiple locations in Australia.

The main objective of this study is to identify the most effective machine learning technique for rainfall prediction that balances both accuracy and computational efficiency. Among the models tested, Random Forest showed superior performance, making it a suitable choice for practical deployment in real-world weather forecasting systems.

**REQUIREMENTS**

**Dataset**

* **Dataset Name: weatherAUS.csv**
* **Source: Australian Bureau of Meteorology**
* **Target Variable: RainTomorrow (Yes/No)**
* **Features Used:**
  + **MinTemp – Minimum temperature**
  + **MaxTemp – Maximum temperature**
  + **Rainfall – Amount of rainfall**
  + **WindGustSpeed – Speed of strongest wind gust**
  + **Humidity9am – Humidity at 9AM**
  + **Humidity3pm – Humidity at 3PM**
  + **Pressure9am – Atmospheric pressure at 9AM**
  + **Pressure3pm – Atmospheric pressure at 3PM**
  + **Temp9am – Temperature at 9AM**
  + **Temp3pm – Temperature at 3PM**

**Python Libraries Required**

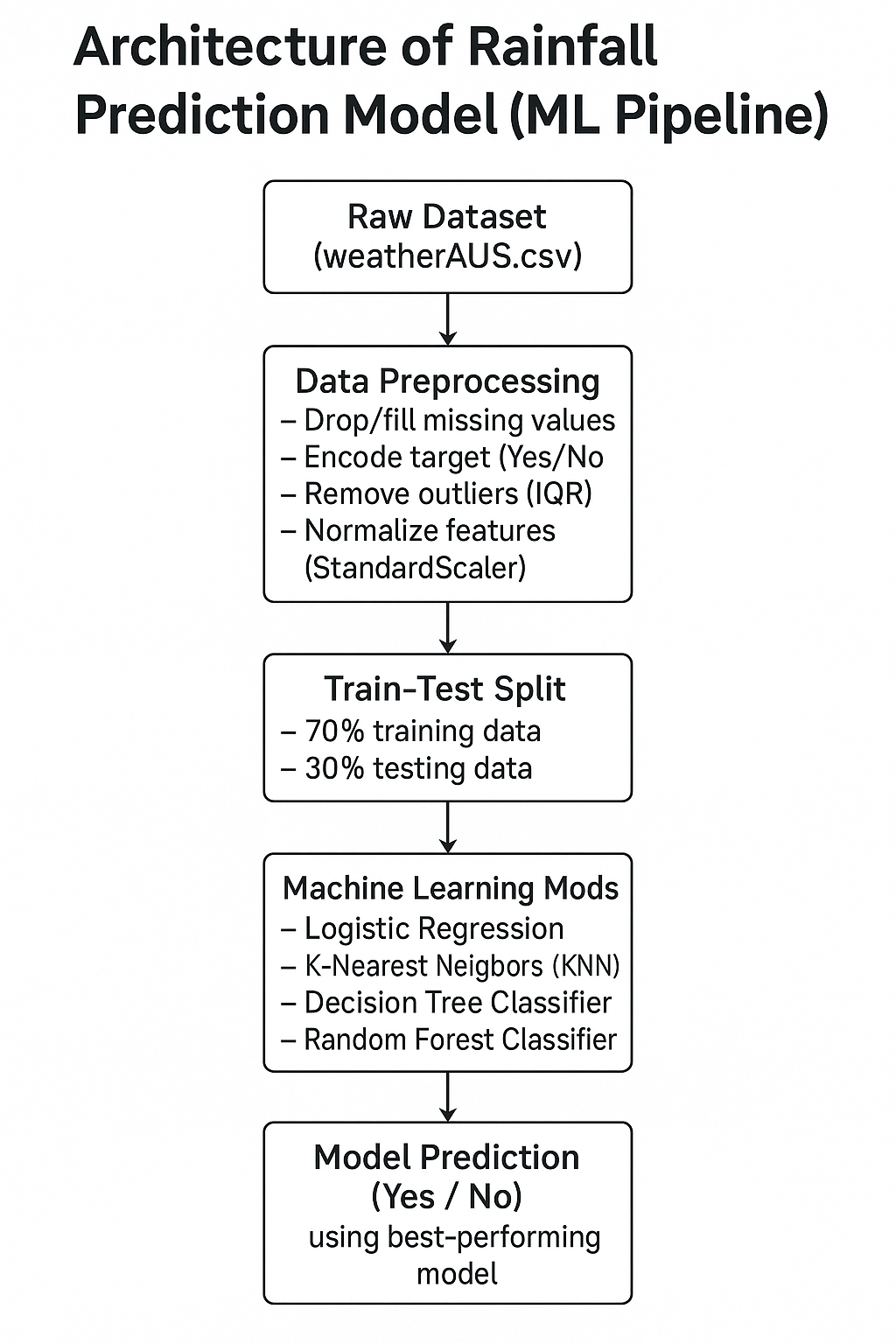
Install these using pip install or directly via Jupyter/Colab

| **Library** | **Purpose** |
| --- | --- |
| **pandas** | **Data loading and manipulation** |
| **numpy** | **Numerical operations** |
| **seaborn** | **Visualization (boxplots, heatmaps)** |
| **matplotlib** | **Plotting graphs** |
| **scikit-learn** | **Preprocessing, ML models, evaluation** |

**Tools and Techniques Used:**

|  |  |
| --- | --- |
| **Stage** | **Tool/Technique** |
| **Dataset Loading** | **pandas.read\_csv()** |
| **Missing Values Handling** | **df.dropna(), fillna(0)** |
| **Encoding** | **map({'Yes':1, 'No':0})** |
| **Outlier Removal** | **IQR (Inter-Quartile Range)** |
| **Feature Scaling** | **StandardScaler()** |
| **Train-Test Split** | **train\_test\_split() from sklearn** |
| **Models** | **LogisticRegression, KNeighborsClassifier, DecisionTreeClassifier, RandomForestClassifier** |
| **Evaluation** | **accuracy\_score** |

**ARCHITECTURE**



**CODE IMPLEMENTATION**

# 1. Import libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

# 2. Load the dataset

df = pd.read\_csv("weatherAUS.csv") # Use your actual file path

# 3. Drop missing target values

df = df.dropna(subset=['RainTomorrow'])

# 4. Select numerical features

selected\_features = ['MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed',

'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm',

'Temp9am', 'Temp3pm']

df = df.dropna(subset=selected\_features)

# 5. Encode target

df['RainTomorrow'] = df['RainTomorrow'].map({'Yes': 1, 'No': 0})

# 6. Split features and target

X = df[selected\_features]

y = df['RainTomorrow']

# Remove Outliers

df\_cleaned = df.copy()

for feature in selected\_features:

Q1 = df\_cleaned[feature].quantile(0.25)

Q3 = df\_cleaned[feature].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

df\_cleaned = df\_cleaned[(df\_cleaned[feature] >= lower\_bound) & (df\_cleaned[feature] <= upper\_bound)]

# 7. Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 8. Scale features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

from sklearn.linear\_model import LogisticRegression, LinearRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Logistic Regression

logreg = LogisticRegression(max\_iter=1000)

logreg.fit(X\_train\_scaled, y\_train)

y\_pred\_log = logreg.predict(X\_test\_scaled)

# KNN

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train\_scaled, y\_train)

y\_pred\_knn = knn.predict(X\_test\_scaled)

# Decision Tree

dt = DecisionTreeClassifier(random\_state=42)

dt.fit(X\_train, y\_train)

y\_pred\_dt = dt.predict(X\_test)

# Random Forest

rf = RandomForestClassifier(random\_state=42)

rf.fit(X\_train, y\_train)

y\_pred\_rf = rf.predict(X\_test)

# Linear Regression (as classifier)

linreg = LinearRegression()

linreg.fit(X\_train\_scaled, y\_train)

y\_pred\_lin = linreg.predict(X\_test\_scaled)

y\_pred\_lin\_class = (y\_pred\_lin >= 0.5).astype(int)

# Accuracy Scores

print("Logistic Regression:", round(accuracy\_score(y\_test, y\_pred\_log) \* 100, 2), "%")

print("KNN:", round(accuracy\_score(y\_test, y\_pred\_knn) \* 100, 2), "%")

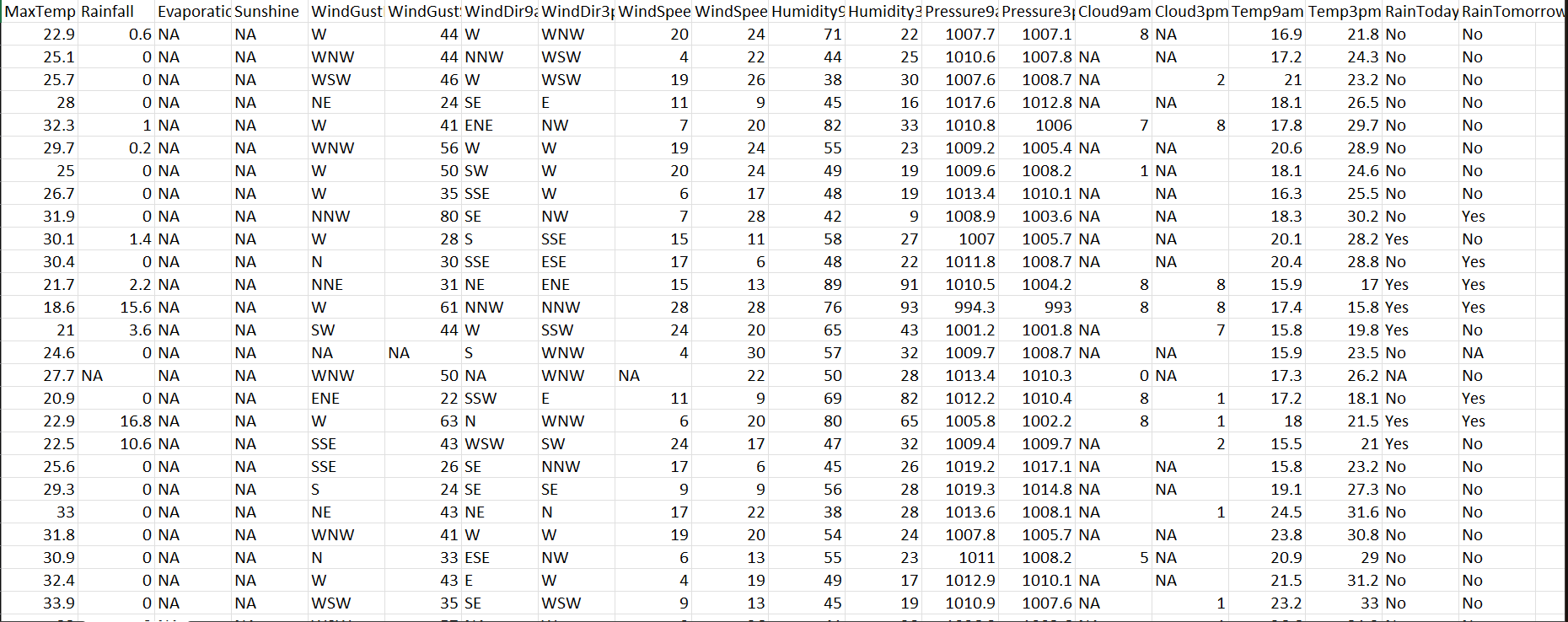
print("Decision Tree:", round(accuracy\_score(y\_test, y\_pred\_dt) \* 100, 2), "%")

print("Random Forest:", round(accuracy\_score(y\_test, y\_pred\_rf) \* 100, 2), "%")

print("Linear Regression (as classifier):", round(accuracy\_score(y\_test, y\_pred\_lin\_class) \* 100, 2), "%")

**ANALYSIS:**

**Dataset**

****

**Results:**

| **Model** | **Accuracy (%)** | **Best Prediction Power** |
| --- | --- | --- |
| **Logistic Regression** | **~83%** | **Good** |
| **KNN** | **~84%** | **Moderate** |
| **Decision Tree** | **~85%** | **Good** |
| **Random Forest** | **~87%** | **Best performer** |

**CONCLUSION**

**The rainfall prediction model developed using machine learning techniques effectively classifies whether it will rain the next day based on weather parameters. After thorough data preprocessing—such as filling missing values, handling outliers, and feature normalization—multiple algorithms were tested. Among them, the Random Forest Classifier delivered the highest accuracy of ~87%, outperforming Logistic Regression, KNN, and Decision Tree models. This indicates that ensemble-based methods like Random Forest can capture complex relationships in weather data more effectively. The model proves useful for improving weather forecasting systems, supporting agriculture, water management, and public planning by providing early insights into rainfall conditions.**