Final Project Report

Rainfall Prediction using Machine Learning

1.Introduction

1.1 Project overviews

Rainfall prediction is an essential component of weather forecasting with wide-ranging implications for agriculture, disaster preparedness, and water resource management. Traditional prediction methods, which rely heavily on meteorological models and historical weather data, often struggle with accuracy due to the complex and nonlinear nature of weather systems. This project leverages machine learning (ML) to enhance rainfall prediction by utilizing vast datasets and sophisticated algorithms, aiming to improve both the precision and reliability of forecasts. By analyzing a range of variables, such as temperature, humidity, wind speed, and geographic features, the ML model can identify patterns and correlations that might be missed by conventional methods, offering a more robust and data-driven approach to predicting rainfall.

1.2 Objectives

The primary objective of this project is to develop an ML model that ca n accurately predict rainfall using historical weather data and other rele vant meteorological variables. Key goals include improving prediction accuracy compared to traditional models, identifying significant pattern s and correlations in weather data, and creating a user-friendly tool for stakeholders in agriculture, water management, and dis aster response. The ultimate aim is to enable these stakeholders to make informed decisions that can mitigate the impacts of weather-related events, optimize agricultural practices, and manage water resour ces more effectively. The project also seeks to advance the field of weat her prediction by demonstrating the potential of ML techniques in captu ring the complexities of weather systems.

2.Project Initialization and Planning Phase

2.1 Define Problem Statements

Farmers and agricultural planners struggle with inaccurate and non-localized weather forecasts, leading to poor planning and potential crop loss. This causes anxiety and uncertainty about the best times to plant and water crops. Similarly, daily commuters and travellers face frustration and disruptions due to untimely and imprecise weather updates, impacting their travel plans and overall experience. Our project aims to address these issues by providing accurate and localized rainfall predictions, helping both groups make informed decisions and improve their productivity and convenience.

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A farmer	Accurately predict rainfall to plan crop planting and irrigation schedules.	leading to poor planning and potential crop loss.	Current weather forecasts are often inaccurate or not localized enough	Anxious and uncertain about the best times to plant and water crops, impacting my livelihood and productivity.
PS-2	Traveler	Plan my travel routes and activities efficiently.	leading to unexpected delays and disruptions.	The weather forecasts are often inaccurate or not timely	The weather forecasts are often inaccurate or not timely

I am	I'm trying to	But	Because	Which makes me feel
• A farmer	• Accurately predict rainfall to plan crop planting and irrigation schedules.	• leading to poor planning and potential crop loss.	• Current weather forecasts are often inaccurate or not localized enough	• Anxious and uncertain about the best times to plant and water crops, impacting my livelihood and productivity.

I am	I'm trying to	But	Because	Which makes me feel
• A Traveler	• Plan my travel routes and activities efficiently.	• leading to unexpected delays and disruptions	• The weather forecasts are often inaccurate or not timely	• The weather forecasts are often inaccurate or not timely

2.2 Project Proposal (Proposed Solution)

This project proposal outlines a solution to address a specific problem. With a clear objective, defined scope, and a concise problem statement, the proposed solution details the approach, key features, and resource requirements, including hardware, software, and personnel.

Project Overview	
Objective	The primary objective of your project is to leverage machine learnin g to accurately predict rainfall patterns. This can help mitigate the i mpact of extreme weather events, support agricultural planning, and enhance water resource management.
Scope	project focuses on predicting rainfall within specific geographical re gions using machine learning models. The extent includes collecting historical weather data, training models, validating accuracy, and im plementing predictions for realtime applications. It doesn't cover oth er weather phenomena like temperature or wind patterns.
Problem Statement	
Description	The project aims to tackle the unpredictable nature of rainfall which can lead to severe consequences like floods, droughts, and crop failu re. By using machine learning, the goal is to provide more accurate a nd timely predictions to better prepare for and mitigate these weathe r-related challenges.
Impact	Nailing this problem with ML could revolutionize how we handle ra infall. Better predictions mean timely disaster management, optimiz ed water resources, and increased agricultural yields. It's a game-changer for food security and climate resilience.
Proposed Solution	
Approach	Employing machine learning techniques to analyze and predict Rainfall, creating a dynamic and adaptable Rainfall prediction System

Key Features	1. This solution harnesses advanced machine learning models for unparalleled rainfall prediction accuracy. 2. It dynamically updates with realtime data, ensuring continuou s adaptability and precision. 3. By incorporating geographical and meteorological variables, it provides a comprehensive approach to understanding rainfall patterns.

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	T4 GPU
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, numpy, matplotlib, seaborn
Development Environment	IDE, version control	Jupyter Notebook, vscode, Git
Data		
Data	Source, size, format	Kaggle dataset, , 690csv, Meteorological departments, open weather datasets

2.3 Initial Project Planning

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Data Collection and Preprocessing	USN-1	Understanding and loading data	2	High	Shaik Mohammad Huzefa	23/09/2024	26/09/2024
Sprint-1	Data Collection and Preprocessing	USN-2	Data cleaning	1	High	Sanka N V Rama Krishna Koundinya	26/09/2024	26/09/2024
Sprint-1	Data Collection and Preprocessing	USN-3	EDA	2	Low	Syed Madhu	23/09/2024	26/09/2024
Sprint-2	Model Development	USN-4	Training the model	2	Medium	Rekha Lokesh	27/09/2024	30/09/2024
Sprint-2	Model tuning and testing	USN-5	Evaluating the model	1	High	Shaik Mohammad Huzefa	27/09/2024	30/09/2024
Sprint-2	Model tuning and testing	USN-6	Model tuning	2	High	Shaik Mohammad Huzefa	27/09/2024	30/09/2024
Sprint-2	Model tuning and testing	USN-7	Model testing	1	Medium	Sanka N V Rama Krishna Koundinya	27/09/2024	30/09/2024
Sprint-3	Web integration and Deployment	USN-8	Building HTML templates	2	Medium	Syed Madhu	01/10/2024	05/10/2024
Sprint-3	Web integration and Deployment	USN-9	Local deployment	2	High	Rekha Lokesh	01/10/2024	05/10/2024
Sprint-4	Project Report	USN-10	Report	2	Medium	Shaik Mohammad Huzefa	06/10/2024	10/10/2024

3. Data Collection and Preprocessing Phase

3.1 Data Collection Plan & Raw Data Sources Identification

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

Data Collection Plan Template

Section	Description
Project Overview	Rainfall prediction using machine learning entails examining histor ical weather data to predict future precipitation. By employing adv anced algorithms such as Decision Trees, Random Forest, and Neu ral Networks, we achieve remarkable accuracy in forecasting rainf all patterns. This significantly supports agricultural planning, water resource management, and disaster preparedness, leading to more informed and effective decision-making.
Data Collection Plan	• Searching for Datasets: Look for datasets related to rainfall occurrence from reliable sources like meteorological departments, online databases (e.g., NOAA, OpenWeatherMap), and research institutions. Prioritize datasets that include comprehensive weather metrics over an extended period. • Prioritize dataset with various demographic information

Raw Data Sources Identified	Gather extensive historical weather data, including temperature, humidity, wind speed, and past rainfall records, from reliable sources like local meteorological stations, national meteorological databases, and online platforms such as NOAA or OpenWeatherMap. Ensure data spans multiple years to capture seasonal and annual variations.

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
Dataset 1	Smart Internz Platform	https://docs.google.com/spreadsheets/d/1RA2OO0LZTeQykI_mvnensAjp6LM4YzWI1Tz0SUG5-Ao/edit?usp=sharing	CSV	13.5 MB	Public
Dataset 2	Kaggle	https://www.kaggle.com/datasets/rajanand/rainfall-in-india	CSV	192 KB	Public

3.2 Data Quality Report

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

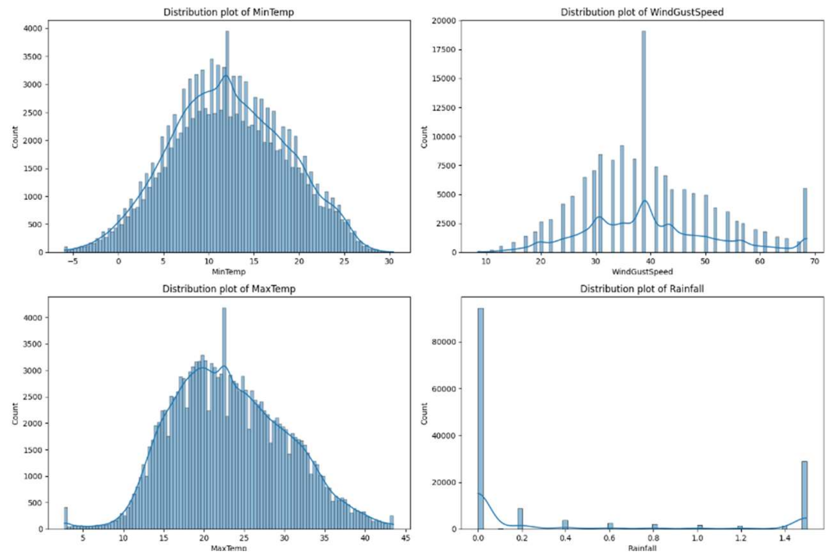
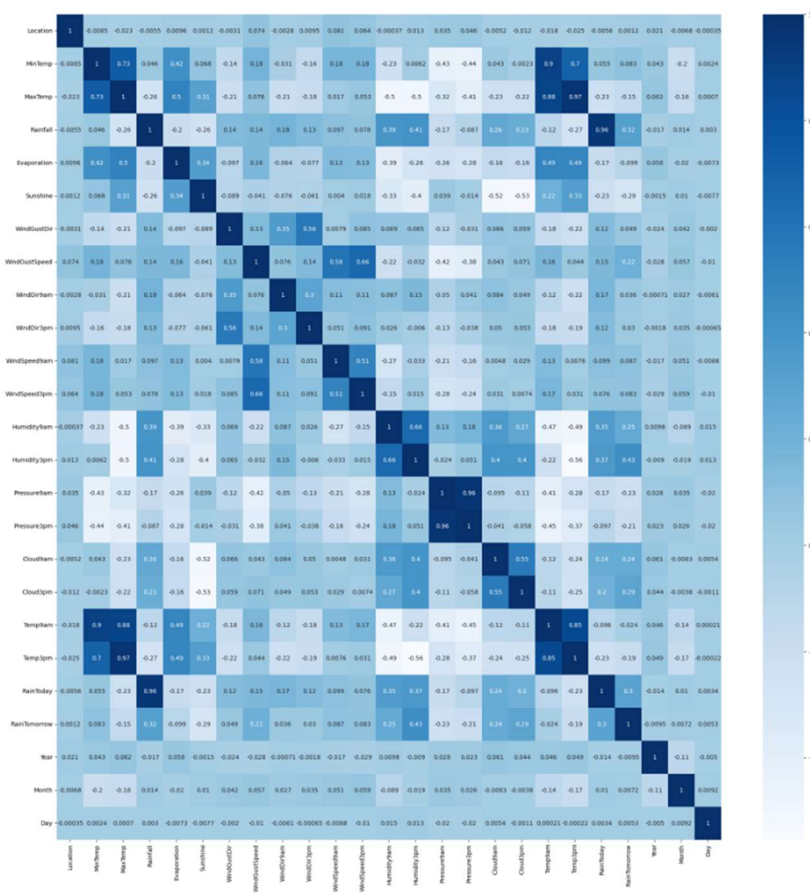
Data Source	**Data Quality Issue**	**Severity**	**Resolution Plan**
Smart Internz Dataset	Missing values in the "MinTemp**","**MaxTemp","Rainfall","Evaporation","Sunshine","WindGustDir","WindGustSpeed","WindDir9am","WindDir3pm","WindSpeed9am","WindSpeed3pm","Humidity9am","Humidity3pm","Pressure9am","Pressure3pm","Cloud9am", "Cloud3pm", "Temp9am**", "**Temp3pm**", "**RainToday", "RainTomorrow"	Moderate	Use mean/mode Imputation

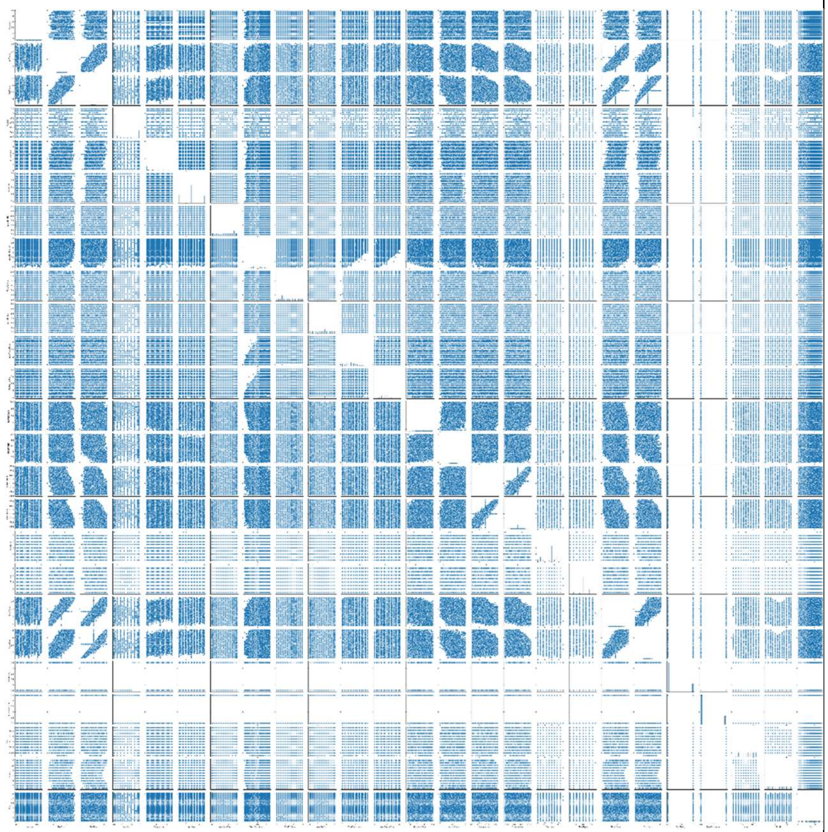
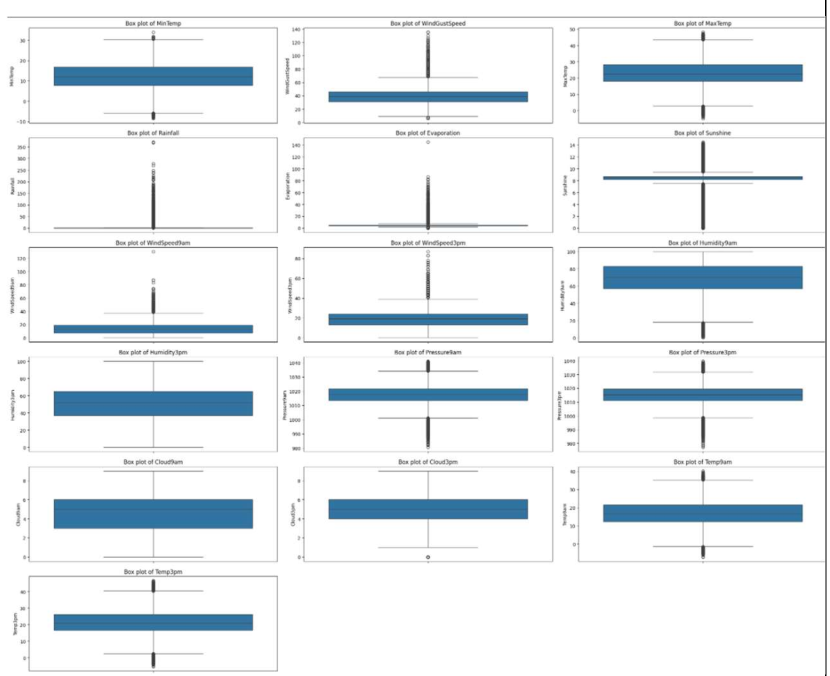
Smart Internz Dataset	Categorical data in the dataset	Moderate	Encoding has to be done in the data

3.3 Data Exploration and Preprocessing Template

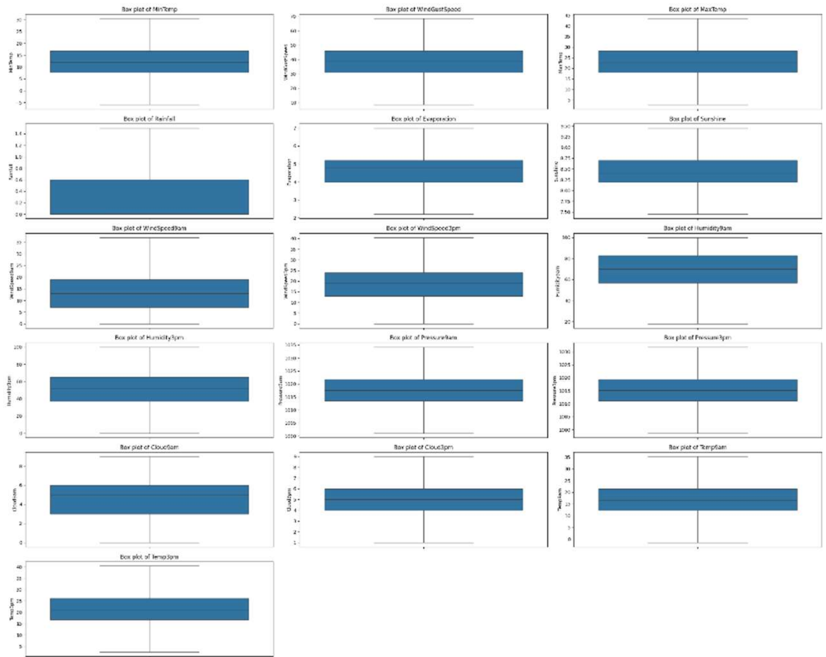
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td>Dimension: 145460 rows × 23 columns
<table>
<tr><th></th><th>MinTemp</th><th>MaxTemp</th><th>Rainfall</th><th>Evaporation</th><th>Sunshine</th><th>WindGustSpeed</th><th>WindSpeed9am</th></tr>
<tr><td>count</td><td>143975.000000</td><td>144199.000000</td><td>142199.000000</td><td>82670.000000</td><td>75625.000000</td><td>135197.000000</td><td>143693.000000</td></tr>
<tr><td>mean</td><td>12.194034</td><td>23.221348</td><td>2.360918</td><td>5.468232</td><td>7.611178</td><td>40.035230</td><td>14.043426</td></tr>
<tr><td>std</td><td>6.398495</td><td>7.119049</td><td>8.478060</td><td>4.193704</td><td>3.785483</td><td>13.607062</td><td>8.915375</td></tr>
<tr><td>min</td><td>-8.500000</td><td>-4.800000</td><td>0.000000</td><td>0.000000</td><td>0.000000</td><td>6.000000</td><td>0.000000</td></tr>
<tr><td>25%</td><td>7.600000</td><td>17.900000</td><td>0.000000</td><td>2.600000</td><td>4.800000</td><td>31.000000</td><td>7.000000</td></tr>
<tr><td>50%</td><td>12.000000</td><td>22.600000</td><td>0.000000</td><td>4.800000</td><td>8.400000</td><td>39.000000</td><td>13.000000</td></tr>
<tr><td>75%</td><td>16.900000</td><td>28.200000</td><td>0.800000</td><td>7.400000</td><td>10.600000</td><td>48.000000</td><td>19.000000</td></tr>
<tr><td>max</td><td>33.900000</td><td>48.100000</td><td>371.000000</td><td>145.000000</td><td>14.500000</td><td>135.000000</td><td>130.000000</td></tr>
</table>
</td></tr>
</table>

Univariate Analysis	
Bivariate Analysis	

Multivariate Analysis	
Outliers and Anomalies	Identification of outliers 

Treatment of outliers.



Data Preprocessing Code Screenshots

Loading Data

```
df=pd.read_csv('/content/Weather.csv - Dataset.csv')
df
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	.
0	2008-12-01	Delhi	13.4	22.9	0.6	NaN	NaN	W	44.0	W	
1	2008-12-02	Delhi	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	
2	2008-12-03	Delhi	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	
3	2008-12-04	Delhi	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	
4	2008-12-05	Delhi	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	
...	...	...	...	...	...	...	...	...	...	...	
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	22.0	SE	
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N	37.0	SE	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE	28.0	SSE	
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	NaN	NaN	NaN	ESE	

145460 rows × 23 columns

Handling Missing Data

Identifying missing values

```
df.isnull().sum()
```

	0
Date	0
Location	0
MinTemp	1485
MaxTemp	1261
Rainfall	3261
Evaporation	62790
Sunshine	69835
WindGustDir	10326
WindGustSpeed	10263
WindDir9am	10566
WindDir3pm	4228
WindSpeed9am	1767
WindSpeed3pm	3062
Humidity9am	2654
Humidity3pm	4507
Pressure9am	15065
Pressure3pm	15028
Cloud9am	55888
Cloud3pm	59358
Temp9am	1767
Temp3pm	3609
RainToday	3261
RainTomorrow	3253

Handling missing values

```
df['MinTemp'].fillna(df['MinTemp'].median(), inplace = True)
df['MaxTemp'].fillna(df['MaxTemp'].median(), inplace = True)
df['Rainfall'].fillna(df['Rainfall'].median(), inplace = True)
df['Evaporation'].fillna(df['Evaporation'].median(), inplace = True)
df['Sunshine'].fillna(df['Sunshine'].median(), inplace = True)
df['WindGustSpeed'].fillna(df['WindGustSpeed'].median(), inplace = True)
df['WindSpeed9am'].fillna(df['WindSpeed9am'].median(), inplace = True)
df['WindSpeed3pm'].fillna(df['WindSpeed3pm'].median(), inplace = True)
df['Humidity9am'].fillna(df['Humidity9am'].median(), inplace = True)
df['Humidity3pm'].fillna(df['Humidity3pm'].median(), inplace = True)
df['Pressure9am'].fillna(df['Pressure9am'].median(), inplace = True)
df['Pressure3pm'].fillna(df['Pressure3pm'].median(), inplace = True)
df['Cloud9am'].fillna(df['Cloud9am'].median(), inplace = True)
df['Cloud3pm'].fillna(df['Cloud3pm'].median(), inplace = True)
df['Temp9am'].fillna(df['Temp9am'].median(), inplace = True)
df['Temp3pm'].fillna(df['Temp3pm'].median(), inplace = True)
```

<table>
<tr>
<td></td>
<td>
<pre><code>df['WindDir9am'].fillna(df['WindDir9am'].mode()[0], inplace=True)
df['WindDir3pm'].fillna(df['WindDir3pm'].mode()[0], inplace=True)
df['RainToday'].fillna(df['RainToday'].mode()[0], inplace=True)
df['RainTomorrow'].fillna(df['RainTomorrow'].mode()[0], inplace=True)
df['WindGustDir'].fillna(df['WindGustDir'].mode()[0], inplace=True)</code></pre>
</td>
</tr>
<tr>
<td>Data Transformation</td>
<td>
Encoding
<pre><code>le = LabelEncoder()

df['WindDir9am'] = le.fit_transform(df['WindDir9am'])
df['WindDir3pm'] = le.fit_transform(df['WindDir3pm'])
df['RainToday'] = le.fit_transform(df['RainToday'])
df['RainTomorrow'] = le.fit_transform(df['RainTomorrow'])
df['Location'] = le.fit_transform(df['Location'])
df['WindGustDir'] = le.fit_transform(df['WindGustDir'])</code></pre>
Scaling
<pre><code>x = df.loc[:, ['Humidity3pm', 'Rainfall', 'Cloud3pm', 'Humidity9am', 'Cloud9am', 'WindGustSpeed', 'WindSpeed9am', 'MinTemp', 'WindSpeed3pm','WindGustDir']]

y = df['RainTomorrow']</code></pre>
Feature Scaling
<pre><code>#scaling the data
sc = StandardScaler()

x_sc = sc.fit_transform(x)

x = pd.DataFrame(x_sc, columns=x.columns)</code></pre>
</td>
</tr>
</table>

<table>
<tr>
<td>Feature Engineering</td>
<td>
<pre><code>df.corr()['RainTomorrow'].sort_values(ascending= False)</code></pre>
<table>
<tr><th></th><th>RainTomorrow</th></tr>
<tr><td>RainTomorrow</td><td>1.000000</td></tr>
<tr><td>Humidity3pm</td><td>0.431272</td></tr>
<tr><td>Rainfall</td><td>0.321671</td></tr>
<tr><td>RainToday</td><td>0.304062</td></tr>
<tr><td>Cloud3pm</td><td>0.290055</td></tr>
<tr><td>Humidity9am</td><td>0.250375</td></tr>
<tr><td>Cloud9am</td><td>0.241909</td></tr>
<tr><td>WindGustSpeed</td><td>0.216257</td></tr>
<tr><td>WindSpeed9am</td><td>0.086720</td></tr>
<tr><td>MinTemp</td><td>0.083237</td></tr>
<tr><td>WindSpeed3pm</td><td>0.082588</td></tr>
<tr><td>WindGustDir</td><td>0.048793</td></tr>
<tr><td>WindDir9am</td><td>0.036326</td></tr>
<tr><td>WindDir3pm</td><td>0.029703</td></tr>
<tr><td>Month</td><td>0.007178</td></tr>
<tr><td>Day</td><td>0.005318</td></tr>
<tr><td>Location</td><td>0.001176</td></tr>
<tr><td>Year</td><td>-0.009535</td></tr>
<tr><td>Temp9am</td><td>-0.023780</td></tr>
<tr><td>Evaporation</td><td>-0.098930</td></tr>
<tr><td>MaxTemp</td><td>-0.154837</td></tr>
<tr><td>Temp3pm</td><td>-0.186139</td></tr>
<tr><td>Pressure3pm</td><td>-0.207057</td></tr>
<tr><td>Pressure9am</td><td>-0.226512</td></tr>
<tr><td>Sunshine</td><td>-0.288945</td></tr>
</table>
Selecting which are highly corelated to the target column and relatable.
</td>
</tr>
<tr>
<td>Save Processed Data</td>
<td>-</td>
</tr>
</table>

4. Model Development Phase

4.1 Feature Selection Report

In the forthcoming update, each feature will be accompanied by a brief description. Users will indicate whether it's selected or not, providing reasoning for their decision. This process will streamline decision-making and enhance transparency in feature selection.

Feature	Description	Selected (Yes/No)	Reasoning
Date	The date of the recorded observation.	No	Lower correlation with target column.
Location	The geographic location of the observation.	No	Explanation of why it was selected or excluded
MinTemp	The minimum temperature recorded for the day	Yes	Influences daily weather predictions.
MaxTemp	The maximum temperature recorded for the day	**No**	Lower correlation with target column.
Rainfall	The amount of rainfall recorded for the day	**Yes**	Direct measure of precipitation.

Evaporation	The amount of evaporation measured for the day	**No**	Lower correlation with target column.
Sunshine	The number of sunshine hours recorded for the day.	**No**	Lower correlation with target column.
WindGustDir	The direction of the strongest wind gust recorded.	**Yes**	Gust direction indicates storm paths**.**
WindGustSpeed	The speed of the strongest wind gust recorded.	**Yes**	Indicates potential for extreme weather.
WindDir9am	The wind direction recorded at 9 AM	**No**	Lower correlation with target column.
WindDir3pm	The wind direction recorded at 3 PM.	**No**	Lower correlation with target column.
WindSpeed9am	The wind speed recorded at 9 AM.	**Yes**	Morning wind patterns influence daily weather.
WindSpeed3pm	The wind speed recorded at 3 PM.	**Yes**	Afternoon wind patterns provide forecasting data.
Humidity9am	The humidity percentage recorded at 9 AM.	**Yes**	Morning humidity influences daily weather.

Humidity3pm	The humidity percentage recorded at 3 PM.	**Yes**	Directly affects precipitation predictions.
Pressure9am	The atmospheric pressure recorded at 9 AM.	**No**	Lower correlation with target column.
Pressure3pm	The atmospheric pressure recorded at 3 PM.	**No**	Lower correlation with target column.
Cloud9am	The cloud cover recorded at 9 AM.	**Yes**	Lower correlation with target column.
Cloud3pm	The cloud cover recorded at 3 PM	**Yes**	Morning cloud cover affects weather outcomes.
Temp9am	The temperature recorded at 9 AM.	**No**	Lower correlation with target column.
Temp3pm	The temperature recorded at 3 PM.	**No**	Lower correlation with target column.
RainToday	Indicates if it rained today.	**No**	High correlation but redundant with Rainfall column already providing relevant data
RainTomorrow	Predicts if it will rain tomorrow.	**Yes**	The target variable for predictive modelling – is essential for project goals.

4.2 Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model Selection Report:

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
XGBoost	Utilizes gradient boosting for efficient, high-performance classification.	-	Accuracy score = 78%
Random Forest Classifier	Builds multiple decision trees for robust predictions.	-	Accuracy score = 83%
Decision Tree Classifier	Uses a tree-like structure for decision making.	-	Accuracy score = **76%**
Gradient Boosting Classifier	Combines weak learners for powerful predictions.	-	Accuracy score = **81%**
Logistic Regression	Employs regression to classify binary targets.	-	Accuracy score = **76%**
K neighbors Classifier	Predicts based on the 'k' nearest data points.	-	Accuracy score = **73%**

4.3 Initial Model Training Code, Model Validation and Evaluation Report

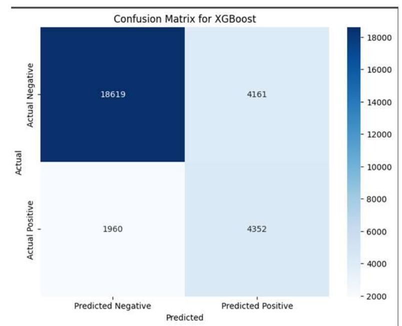
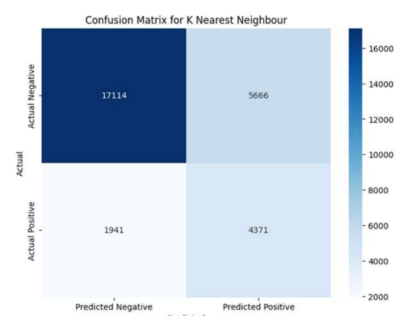
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

Paste the screenshot of the model training code

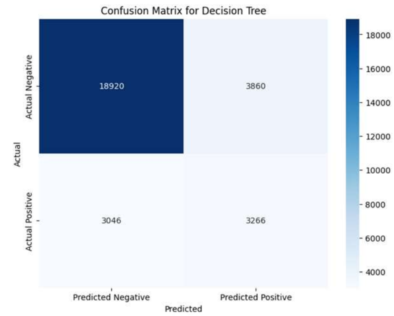
```
XGBoost = xgboost.XGBRFClassifier()
Rand_forest = RandomForestClassifier()
Dtree = DecisionTreeClassifier()
GBM = GradientBoostingClassifier()
log = LogisticRegression()
Knn = KNeighborsClassifier()

XGBoost.fit(x_train_smote, y_train_smote)
Rand_forest.fit(x_train_smote, y_train_smote)
Dtree.fit(x_train_smote, y_train_smote)
GBM.fit(x_train_smote, y_train_smote)
log.fit(x_train_smote, y_train_smote)
Knn.fit(x_train_smote,y_train_smote)
```

Model Validation and Evaluation Report:

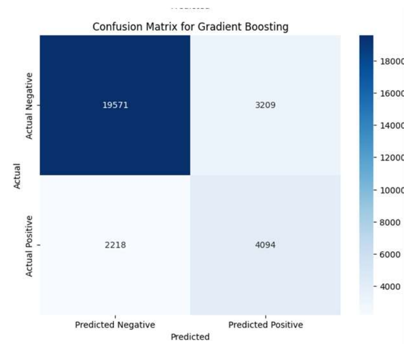
Model	Classification Report	Accuracy	Confusion Matrix
XGBoost	Classification Report for XGBoost: precision recall f1-score support 0 0.90 0.82 0.86 22780 1 0.51 0.69 0.59 6312 accuracy 0.79 29092 macro avg 0.71 0.75 0.72 29092 weighted avg 0.82 0.79 0.80 29092	78%	
Random Forest	Classification Report for Random Forest: precision recall f1-score support 0 0.88 0.92 0.90 22780 1 0.65 0.55 0.60 6312 accuracy 0.84 29092 macro avg 0.76 0.74 0.75 29092 weighted avg 0.83 0.84 0.83 29092	83%	
K Nearest Neighbour	Classification Report for K Nearest Neighbour: precision recall f1-score support 0 0.90 0.75 0.82 22780 1 0.44 0.69 0.53 6312 accuracy 0.74 29092 macro avg 0.67 0.72 0.68 29092 weighted avg 0.80 0.74 0.76 29092	73%	

<table>
<tr>
<td>Decision Tree</td>
<td><pre>
Classification Report for Decision Tree:
              precision   recall f1-score   support

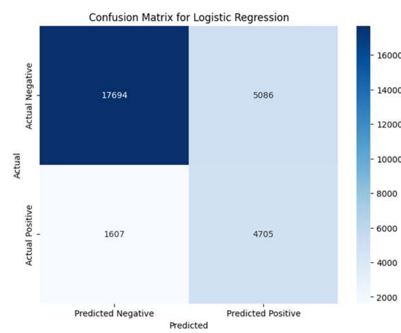
           0   0.86   0.83   0.85   22780
           1   0.46   0.52   0.49   6312

    accuracy   0.76   29092
   macro avg   0.66   0.67   0.67   29092
weighted avg   0.77   0.76   0.77   29092
</pre></td>
<td>76%</td>
<td>
</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td><pre>
Classification Report for Gradient Boosting:
              precision   recall f1-score   support

           0   0.90   0.86   0.88   22780
           1   0.56   0.65   0.60   6312

    accuracy   0.81   29092
   macro avg   0.73   0.75   0.74   29092
weighted avg   0.82   0.81   0.82   29092
</pre></td>
<td>81%</td>
<td>
</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td><pre>
Classification Report for Logistic Regression:
              precision   recall f1-score   support

           0   0.92   0.78   0.84   22780
           1   0.48   0.75   0.58   6312

    accuracy   0.77   29092
   macro avg   0.70   0.76   0.71   29092
weighted avg   0.82   0.77   0.79   29092
</pre></td>
<td>76%</td>
<td>
</td>
</tr>
</table>

5. Model Optimization and Tuning Phase Template

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

5.1 Hyperparameter Tuning Documentation

Model	Tuned Hyperparameters	Optimal Values

XGBoost	`"XGBoost": {`
	`'n_estimators': [100, 200],`
	`'learning_rate': [0.01, 0.1],`
	`'max_depth': [3, 5, 7]`
	`},`
		`'XGBoost': xgboost.XGBRFClassifier(base_score=None, booster=None, callbacks=None,`
		`colsample_bylevel=None, colsample_bynode=None,`
		`colsample_bytree=None, device=None, early_stopping_rounds=None,`
		`enable_categorical=False, eval_metric=None, feature_types=None,`
		`gamma=None, grow_policy=None, importance_type=None,`
		`interaction_constraints=None, learning_rate=0.1, max_bin=None,`
		`max_cat_threshold=None, max_cat_to_onehot=None,`
		`max_delta_step=None, max_depth=7, max_leaves=None,`
		`min_child_weight=None, monotone_constraints=None,`
		`multi_strategy=None, n_estimators=200, n_jobs=None,`
		`num_parallel_tree=None, random_state=None),`

Random Forest Classifier	`"Random Forest": {`
	`'n_estimators': [100, 200],`
	`'max_depth': [None, 10, 20],`
	`'min_samples_split': [2, 5]`
	`},`
		`RandomForestClassifier(n_estimators=200),`

Decision Tree Classifier	`"Decision Tree": {`
	`'max_depth': [None, 10, 20],`
	`'min_samples_split': [2, 5]`
	`},`
		`Tuning hyperparameters for Decision Tree...`
		`Best parameters for Decision Tree: {'max_depth': None, 'min_samples_split': 2`

Gradient Boosting Classifier	`"Gradient Boosting": {`
	`'n_estimators': [100, 200],`
	`'learning_rate': [0.01, 0.1],`
	`'max_depth': [3, 5, 7]`
	`},`
		`GradientBoostingClassifier(max_depth=7, n_estimators=200),`

Logistic Regression	`"Logistic Regression": {`
	`'C': [0.1, 1, 10],`
	`'penalty': ['l2'],`
	`'solver': ['lbfgs']`
	`}`
		`Logistic Regression: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}`

K Nearest Neighbour	`"K Nearest Neighbour": {`
	`'n_neighbors': [3, 5, 7],`
	`'weights': ['uniform', 'distance']`
	`},`
		`KNeighborsClassifier(n_neighbors=3, weights='distance').`

5.2 Performance Metrics Comparison Report

Model	Baseline Metric	Optimized Metric

XGBoost	78%	`Evaluating XGBoost...`
		`Classification Report:`
		`precision recall f1-score support`
		`0 0.91 0.81 0.86 22780`
		`1 0.51 0.69 0.59 6312`
		`accuracy 0.79 29092`
		`macro avg 0.71 0.75 0.72 29092`
		`weighted avg 0.82 0.79 0.80 29092`
		`Confusion Matrix:`
		`[[18518 4262]`
		`[ 1926 4386]]`

<table>
<tr><td>Random Forest Classifier</td><td>83%</td><td>Evaluating Random Forest...
Classification Report:
precision recall f1-score support

0 0.88 0.92 0.90 22780
1 0.65 0.56 0.60 6312

accuracy 0.84 29092
macro avg 0.77 0.74 0.75 29092
weighted avg 0.83 0.84 0.83 29092

Confusion Matrix:
[[20881 1899]
[2795 3517]]</td></tr>
<tr><td>Decision Tree Classifier</td><td>76%</td><td>Evaluating Decision Tree...
Classification Report:
precision recall f1-score support

0 0.86 0.83 0.85 22780
1 0.46 0.52 0.49 6312

accuracy 0.76 29092
macro avg 0.66 0.67 0.67 29092
weighted avg 0.77 0.76 0.77 29092

Confusion Matrix:
[[18895 3885]
[3043 3269]]</td></tr>
<tr><td>Gradient Boosting Classifier</td><td>81%</td><td>Evaluating Gradient Boosting...
Classification Report:
precision recall f1-score support

0 0.87 0.94 0.91 22780
1 0.70 0.51 0.59 6312

accuracy 0.85 29092
macro avg 0.79 0.72 0.75 29092
weighted avg 0.84 0.85 0.84 29092

Confusion Matrix:
[[21400 1380]
[3091 3221]]</td></tr>
<tr><td>Logistic Regression</td><td>76%</td><td>Evaluating Logistic Regression...
Classification Report:
precision recall f1-score support

0 0.92 0.78 0.84 22780
1 0.48 0.75 0.58 6312

accuracy 0.77 29092
macro avg 0.70 0.76 0.71 29092
weighted avg 0.82 0.77 0.79 29092

Confusion Matrix:
[[17693 5087]
[1606 4706]]</td></tr>
</table>

K Nearest Neighbour	73%	Evaluating K Nearest Neighbour... Classification Report: precision recall f1-score support 0 0.89 0.77 0.83 22780 1 0.44 0.66 0.53 6312 accuracy 0.75 29092 macro avg 0.67 0.71 0.68 29092 weighted avg 0.79 0.75 0.76 29092 Confusion Matrix: [[17561 5219] [2167 4145]]

5.3 Final Model Selection Justification

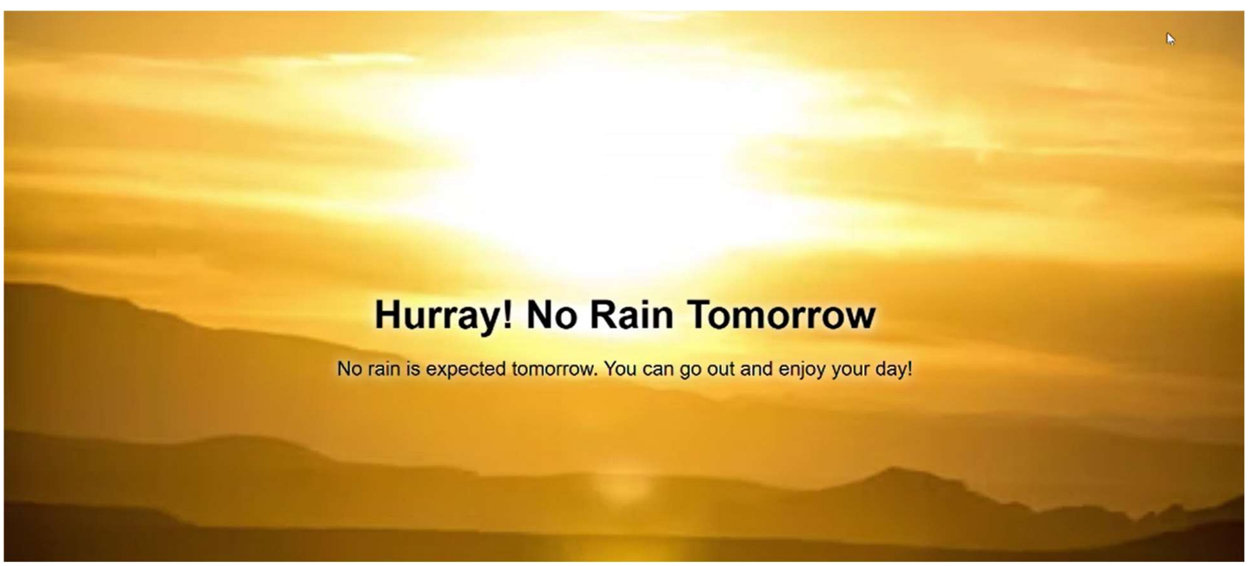
Final Model	Reasoning
Gradient Boosting Classifier	The Gradient Boosting Classifier model was selected for its superior performance, exhibiting high accuracy(85%) during hyperparameter tuning, Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model

6. Results

6.1 Outputs screenshots

Output for there will be rain tomorrow

Output for there will be no rain tomorrow



7. Advantages & Disadvantages

Advantages

1. **Accuracy**: Machine learning models can achieve high accuracy in rainfall prediction, which is crucial for planning and preparedness.
2. **Automation**: Once trained, ML models can automatically process data and provide real-time predictions without manual intervention.
3. **Scalability**: Machine learning models can handle large datasets and adapt to new data, making them scalable as more data becomes available.
4. **Pattern Recognition**: ML models excel at identifying complex patterns in data that might not be apparent with traditional statistical methods.
5. **Customization**: Models can be tailored to specific geographical areas, improving the relevance and accuracy of predictions.
6. **Cost-Effective**: Over time, automated models can reduce the need for extensive human analysis, potentially lowering operational costs.

Disadvantages

1. **Data Dependency**: High-quality, extensive datasets are crucial for training accurate models. Insufficient or poor-quality data can significantly impact model performance.
2. **Complexity**: Developing, training, and fine-tuning machine learning models can be complex and requires expertise in the field.
3. **Resource Intensive**: Training machine learning models can be resource-intensive, requiring substantial computational power and time.
4. **Model Maintenance**: Models need to be continuously updated with new data to maintain accuracy over time, requiring ongoing maintenance efforts.
5. **Interpretability**: ML models, especially deep learning models, can be seen as black boxes, making it challenging to understand how predictions are generated.
6. **Bias**: If the training data is biased, the model may produce biased predictions, affecting the reliability of the results.

8. Conclusion

The application of gradient boosting in our rainfall detection project has illustrated the power and potential of machine learning techniques in the field of meteorology. Gradient boosting, a powerful ensemble method, combines multiple weak learners to create a strong predictive model, thereby improving the overall accuracy and robustness of our rainfall predictions.

In this project, we meticulously preprocessed the dataset to ensure high-quality inputs, which is critical for the performance of any machine learning model. The gradient boosting algorithm was then employed to capture the complex patterns in the data. This method excels at handling non-linear relationships, which are often present in meteorological data. By iteratively correcting the errors of preceding models, gradient boosting fine-tuned the predictions, resulting in a highly accurate model.

Our evaluation metrics, including accuracy, precision, and recall, demonstrated the effectiveness of the gradient boosting model. The accuracy metric indicated how well the model was able to predict rainfall correctly, while precision and recall provided insights into the model's performance in detecting true positive and true negative instances of rainfall. These metrics are crucial for applications that rely on precise weather predictions, such as agriculture, urban planning, and disaster management.

One of the significant advantages of using gradient boosting is its flexibility and adaptability. The model can be easily retrained and updated with new data, ensuring that it remains accurate over time as new weather patterns emerge. Moreover, its ability to handle missing data and noisy inputs makes it particularly suitable for real-world datasets, which often come with such imperfections.

Despite these advantages, it is essential to acknowledge the limitations of our approach. Gradient boosting models can be computationally intensive, requiring significant processing power and time for training, especially with large datasets. Additionally, they are prone to overfitting if not properly tuned, which can lead to decreased performance on unseen data. Therefore, careful consideration of model parameters and regular validation are necessary to maintain the model's accuracy.

In conclusion, our use of gradient boosting for rainfall detection has proven to be a valuable tool in enhancing prediction accuracy and reliability. The insights gained from this project can pave the way for further advancements in weather prediction models, contributing to better preparedness and response strategies in various sectors. By continuing to refine our models and incorporating more diverse datasets, we can unlock even greater potential in machine learning applications for meteorology, ultimately leading to more informed and timely decisions in the face of changing weather patterns.

9. Future Scope

1. Model Enhancement: Continuously refine the model with more diverse and extensive datasets to improve accuracy and robustness.
2. Integration with IoT: Incorporate real-time data from IoT devices, such as weather stations and sensors, to provide up-to-date predictions and enable real-time monitoring.
3. User-Friendly Interface: Develop a user-friendly web or mobile application that presents predictions in an easily understandable format for end-users.
4. Geographic Expansion: Adapt and expand the model to cover different geographic regions, taking into account local climate variations and data availability.
5. Climate Change Impact: Study the impact of climate change on rainfall patterns and incorporate these findings into the model to enhance long-term prediction accuracy.
6. Collaboration with Experts: Work with meteorologists and domain experts to continually validate and improve the model, ensuring it stays relevant and accurate.

10. Appendix

10.1 Source Code

Rainfall prediction.ipynb

```
#IMPORTING NECESSARY LIBRARIES

""""

import pandas as pd

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score
```

```
from sklearn.neighbors import KNeighborsClassifier

import xgboost

from sklearn.metrics import accuracy_score, classification_report

from sklearn.model_selection import KFold, cross_val_score

from sklearn.metrics import confusion_matrix, classification_report

from sklearn.metrics import roc_curve, auc, RocCurveDisplay

from sklearn.model_selection import GridSearchCV

import pickle

from imblearn.over_sampling import SMOTE


"""#Importing the Dataset"""


df=pd.read_csv('/content/Weather.csv - Dataset.csv')

df


"""#Analysing the data"""


df.head()


df.describe()


df.info()
```

```
"""#Handling Missing Values"""

df.isnull().sum()

df['MinTemp'].fillna(df['MinTemp'].median(), inplace = True)
df['MaxTemp'].fillna(df['MaxTemp'].median(), inplace = True)
df['Rainfall'].fillna(df['Rainfall'].median(), inplace = True)
df['Evaporation'].fillna(df['Evaporation'].median(), inplace = True)
df['Sunshine'].fillna(df['Sunshine'].median(), inplace = True)
df['WindGustSpeed'].fillna(df['WindGustSpeed'].median(), inplace = True)
df['WindSpeed9am'].fillna(df['WindSpeed9am'].median(), inplace = True)
df['WindSpeed3pm'].fillna(df['WindSpeed3pm'].median(), inplace = True)
df['Humidity9am'].fillna(df['Humidity9am'].median(), inplace = True)
df['Humidity3pm'].fillna(df['Humidity3pm'].median(), inplace = True)
df['Pressure9am'].fillna(df['Pressure9am'].median(), inplace = True)
df['Pressure3pm'].fillna(df['Pressure3pm'].median(), inplace = True)
df['Cloud9am'].fillna(df['Cloud9am'].median(), inplace = True)
df['Cloud3pm'].fillna(df['Cloud3pm'].median(), inplace = True)
df['Temp9am'].fillna(df['Temp9am'].median(), inplace = True)
df['Temp3pm'].fillna(df['Temp3pm'].median(), inplace = True)
```

```
df['WindDir9am'].fillna(df['WindDir9am'].mode()[0], inplace=True)


df['WindDir3pm'].fillna(df['WindDir3pm'].mode()[0], inplace=True)

df['RainToday'].fillna(df['RainToday'].mode()[0], inplace=True)

df['RainTomorrow'].fillna(df['RainTomorrow'].mode()[0], inplace=True)

df['WindGustDir'].fillna(df['WindGustDir'].mode()[0], inplace=True)


df.isnull().sum()


df


"""#Handling Outliers"""


def plot_box_plots(df, columns):

    plt.figure(figsize=(25, 20))

    for i, column in enumerate(columns, 1):

        plt.subplot(len(columns) // 3 + 1, 3, i)

        sns.boxplot(y=df[column])

        plt.title(f'Box plot of {column}')

    plt.tight_layout()

    plt.show()
```

```
numerical_columns = ['MinTemp','WindGustSpeed', 'MaxTemp', 'Rainfall',
'Evaporation', 'Sunshine', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',

'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
'Temp3pm']

plot_box_plots(df, numerical_columns)

def handle_outliers_iqr(column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Replace outliers with the nearest non-outlier values
    df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
    df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])

for column in numerical_columns:
    handle_outliers_iqr(column)

plot_box_plots(df, numerical_columns)
```

```
"""#Encoding the categorical columns"""

le = LabelEncoder()

df['WindDir9am'] = le.fit_transform(df['WindDir9am'])
df['WindDir3pm'] = le.fit_transform(df['WindDir3pm'])
df['RainToday'] = le.fit_transform(df['RainToday'])
df['RainTomorrow'] = le.fit_transform(df['RainTomorrow'])
df['Location'] = le.fit_transform(df['Location'])
df['WindGustDir'] = le.fit_transform(df['WindGustDir'])

wind_gust_dir_mapping = dict(zip(le.classes_, le.transform(le.classes_)))

print(wind_gust_dir_mapping)

# Convert the 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Extract year, month, and day into new columns
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
```

```
  df['Day'] = df['Date'].dt.day
# Drop the original 'Date' column
  df = df.drop(columns=['Date'])


  df


  df.info()


  """#Data Visualization"""


  df.boxplot()


  def plot_dist_plots(df, columns):
      num_columns = len(columns)
      rows = (num_columns // 2) + (num_columns % 2)  # Adjust the number of rows
      plt.figure(figsize=(15, rows * 5))  # Increase the height of the figure
      for i, column in enumerate(columns, 1):
          plt.subplot(rows, 2, i)  # Use 2 columns for subplots
          sns.histplot(df[column], kde=True)  # Use histplot with KDE
          plt.title(f'Distribution plot of {column}')
      plt.tight_layout()
      plt.show()
```

```
plot_dist_plots(df, numerical_columns)

sns.pairplot(df)

df.corr()

plt.figure(figsize=(25,25))
sns.heatmap(df.corr(),annot=True,cmap='Blues')

"""#Splitting

#Checking corelation according to RainTomorrow column
"""

df.corr()['RainTomorrow'].sort_values(ascending= False)

"""#Splitting the Dateset into Dependent and Independent variable"""

x = df.loc[:, ['Humidity3pm', 'Rainfall', 'Cloud3pm', 'Humidity9am', 'Cloud9am',
'WindGustSpeed', 'WindSpeed9am', 'MinTemp', 'WindSpeed3pm','WindGustDir']]
```

```
y = df['RainTomorrow']


"""#Feature Scaling"""


#scaling the data

sc = StandardScaler()


x_sc = sc.fit_transform(x)


x = pd.DataFrame(x_sc, columns=x.columns)


with open('scaler.pkl', 'wb') as f:

    pickle.dump(sc, f)


x.head()


"""#Splitting the data into Train and Test"""


x_train,x_test,y_train,y_test =train_test_split(x,y,test_size =0.2,random_state = 0)


def calculate_baseline_metric(y_test):

    # Calculate the majority class
```

```
    majority_class = max(np.bincount(y_test))
    baseline_accuracy = majority_class / len(y_test)
    return baseline_accuracy

baseline_accuracy = calculate_baseline_metric(y_test)
print("Baseline metric (accuracy):", baseline_accuracy)

"""#Training and Testing the Model"""

#balancing the data
smote = SMOTE()
x_train_smote, y_train_smote = smote.fit_resample(x_train,y_train)

y_train.value_counts()

y_train_smote.value_counts()

XGBoost = xgboost.XGBRFClassifier()
Rand_forest = RandomForestClassifier()
Dtree = DecisionTreeClassifier()
GBM = GradientBoostingClassifier()
```

```
log = LogisticRegression()

Knn = KNeighborsClassifier()


XGBoost.fit(x_train_smote, y_train_smote)

Rand_forest.fit(x_train_smote, y_train_smote)

Dtree.fit(x_train_smote, y_train_smote)

GBM.fit(x_train_smote, y_train_smote)

log.fit(x_train_smote, y_train_smote)

Knn.fit(x_train_smote,y_train_smote)


pred1 = XGBoost.predict(x_train_smote)

pred2 = Rand_forest.predict(x_train_smote)

pred3 = Knn.predict(x_train_smote)

pred4 = Dtree.predict(x_train_smote)

pred5 = GBM.predict(x_train_smote)

pred6 = log.predict(x_train_smote)


"""#Model Evaluation"""


#Accuracy_score

print("XGBoost:", accuracy_score(y_train_smote, pred1))
```

```
print("Random Forest:", accuracy_score(y_train_smote, pred2))

print("K Nearest Neighbour", accuracy_score(y_train_smote,pred3))

print("Decision Tree:", accuracy_score(y_train_smote, pred4))

print("Gradient Boosting:", accuracy_score(y_train_smote, pred5))

print("Logistic Regression:", accuracy_score(y_train_smote, pred6))


ypred1 = XGBoost.predict(x_test)

ypred2 = Rand_forest.predict(x_test)

ypred3 = Knn.predict(x_test)

ypred4 = Dtree.predict(x_test)

ypred5 = GBM.predict(x_test)

ypred6 = log.predict(x_test)


print("XGBoost:", accuracy_score(y_test, ypred1))

print("Random Forest:", accuracy_score(y_test, ypred2))

print("K Nearest Neighbour", accuracy_score(y_test,ypred3))

print("Decision Tree:", accuracy_score(y_test, ypred4))

print("Gradient Boosting:", accuracy_score(y_test, ypred5))

print("Logistic Regression:", accuracy_score(y_test, ypred6))


#  k-fold cross-validation
```

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)



print("XGBoost:", cross_val_score(XGBoost, x, y, cv=kf).mean())
print("Random Forest:", cross_val_score(Rand_forest, x, y, cv=kf).mean())
print("K Nearest Neighbour", cross_val_score(Knn, x, y, cv=kf).mean())
print("Decision Tree:", cross_val_score(Dtree, x, y, cv=kf).mean())
print("Gradient Boosting:", cross_val_score(GBM, x, y, cv=kf).mean())
print("Logistic Regression:", cross_val_score(log, x, y, cv=kf).mean())


models = ["XGBoost", "Random Forest", "K Nearest Neighbour", "Decision Tree", "Gradient Boosting", "Logistic Regression"]
predictions = [ypred1, ypred2, ypred3, ypred4, ypred5, ypred6]


# Function to plot confusion matrix
def plot_confusion_matrix(y_test, y_pred, model_name):
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Predicted Negative', 'Predicted Positive'], yticklabels=['Actual Negative', 'Actual Positive'])
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted')
```

```
    plt.ylabel('Actual')
    plt.show()

# Loop through each model and its predictions
for model, y_pred in zip(models, predictions):
    plot_confusion_matrix(y_test, y_pred, model)

#crosstable
def create_crosstab(y_test, y_pred, model_name):
    ct = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])
    print(f'Crosstab for {model_name}:\n{ct}\n')

# Function to print classification report
def print_classification_report(y_test, y_pred, model_name):
    report = classification_report(y_test, y_pred)
    print(f'Classification Report for {model_name}:\n{report}\n')

# Loop through each model and its predictions
for model, y_pred in zip(models, predictions):
    create_crosstab(y_test, y_pred, model)
    print_classification_report(y_test, y_pred, model)
```

```
# Roc-Auc Curve
def plot_roc_auc(y_test, y_pred, model_name):
    fpr, tpr, _ = roc_curve(y_test, y_pred)
    roc_auc = auc(fpr, tpr)
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='grey', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC-AUC Curve for {model_name}')
    plt.legend(loc="lower right")
    plt.show()

# Loop through each model and its predictions
for model, y_pred in zip(models, predictions):
    plot_roc_auc(y_test, y_pred, model)

XGBoost = xgboost.XGBRFClassifier()
Rand_forest = RandomForestClassifier()
```

```
svm = SVC()

Dtree = DecisionTreeClassifier()

GBM = GradientBoostingClassifier()

log = LogisticRegression()

Knn = KNeighborsClassifier()


"""#Checking Hyper parameters"""


def hyperparameter_tuning(models, param_grids, x_train_smote, y_train_smote):
    best_models = {}

    for model_name, model in models.items():
        print(f"Tuning hyperparameters for {model_name}...")
        grid_search = GridSearchCV(estimator=model,
param_grid=param_grids[model_name], cv=5, scoring='accuracy', n_jobs=-1)
        grid_search.fit(x_train_smote, y_train_smote)
        best_models[model_name] = grid_search.best_estimator_
        print(f"Best parameters for {model_name}: {grid_search.best_params_}")

    return best_models
```

```
# Example usage:

modelH = {

    "XGBoost": xgboost.XGBClassifier(),

    "Random Forest": RandomForestClassifier(),

    "K Nearest Neighbour": KNeighborsClassifier(),

    "Decision Tree": DecisionTreeClassifier(),

    "Gradient Boosting": GradientBoostingClassifier(),

    "Logistic Regression": LogisticRegression()

}


param_grids = {

    "XGBoost": {

        'n_estimators': [100, 200],

        'learning_rate': [0.01, 0.1],

        'max_depth': [3, 5, 7]

    },

    "Random Forest": {

        'n_estimators': [100, 200],

        'max_depth': [None, 10, 20],

        'min_samples_split': [2, 5]

    },
```

```
 "K Nearest Neighbour": {
        'n_neighbors': [3, 5, 7],
        'weights': ['uniform', 'distance']
    },
    "Decision Tree": {
        'max_depth': [None, 10, 20],
        'min_samples_split': [2, 5]
    },
    "Gradient Boosting": {
        'n_estimators': [100, 200],
        'learning_rate': [0.01, 0.1],
        'max_depth': [3, 5, 7]
    },
    "Logistic Regression": {
        'C': [0.1, 1, 10],
        'penalty': ['l2'],
        'solver': ['lbfgs']
    }
}

# Assuming X_train and y_train are your training data
```

```
best_models = hyperparameter_tuning(modelH, param_grids, x_train_smote,
y_train_smote)


best_models = {
   'XGBoost': xgboost.XGBRFClassifier(base_score=None, booster=None,
callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=7, max_leaves=None,
              min_child_weight=None, monotone_constraints=None,
              multi_strategy=None, n_estimators=200, n_jobs=None,
              num_parallel_tree=None, random_state=None),
 'Random Forest': RandomForestClassifier(n_estimators=200),
 'K Nearest Neighbour': KNeighborsClassifier(n_neighbors=3, weights='distance'),
 'Decision Tree': DecisionTreeClassifier(),
 'Gradient Boosting': GradientBoostingClassifier(max_depth=7,
n_estimators=200),
 'Logistic Regression': LogisticRegression(C=10)
```

```
}


best_scores = {}
for model_name, model in best_models.items():
    y_pred = model.predict(x_train_smote)
    score = accuracy_score(y_train_smote, y_pred)
    best_scores[model_name] = score
    print(f"Best score for {model_name}: {score}")


best_scores = {}
for model_name, model in best_models.items():
    y_pred = model.predict(x_test)
    score = accuracy_score(y_test, y_pred)
    best_scores[model_name] = score
    print(f"Best score for {model_name}: {score}")


def fit_and_evaluate_models(models, X_train, y_train, X_test, y_test):
    for name, model in models.items():
        print(f"Fitting {name}...")
        model.fit(X_train, y_train)
        print(f"Evaluating {name}...")
        y_pred = model.predict(X_test)
```

```
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("\n")

fit_and_evaluate_models(best_models, x_train_smote, y_train_smote, x_test,
y_test)

"""Ater checking the performaces of all models after Evaluation
GradientBoostingClassifier is giving good results so choosing
GradientBoostingClassifier as our final models"""

x_train_smote.head()

y_train_smote.head()

y_test.head()

x_test.head()

pres_model = GradientBoostingClassifier(max_depth=5, n_estimators=200)
```

```
pres_model.fit(x_train_smote, y_train_smote)

y_predict = pres_model.predict(x_test)

accuracy_score(y_test,y_predict)

pres_model.predict([[65, 5.2, 6, 80, 5, 45, 20, 15, 25, 7]])

with open('RFmodel.pkl', 'wb') as file:
    pickle.dump(pres_model, file)

with open('RFmodel.pkl', 'rb') as file:
    loaded_model = pickle.load(file)

loaded_model.predict([[65, 5.2, 6, 80, 5, 45, 20, 15, 25, 7]])

loaded_model.predict([[-0.027033, -0.298494, 0.125622, -0.050953, 0.149133,
0.850594, 1.643330, -1.114363, 0.404729, 1]])

loaded_model.predict([[65, 5.2, 6, 80, 5, 45, 20, 15, 25, 7]])
```

loaded_model.predict([[0.021816, -0.627098, 0.125622, 1.018332, 0.149133, -2.023944, -1.395492, -0.737270, -1.372022, 0]])

app.py

```
# app.py
from flask import Flask, render_template, request
import pickle
import pandas as pd

def create_app():
    app = Flask(__name__)

    # Load the model
    model = pickle.load(open('RFmodel.pkl', 'rb'))
    scaler = pickle.load(open('scaler.pkl', 'rb'))

    @app.route('/')
    def home():
        return render_template('index.html')
```

```
@app.route('/predict', methods=['POST'])
def predict():
# Get data from form in the specified order
    humidity_3pm = float(request.form['Humidity3pm'])
    rainfall = float(request.form['Rainfall'])
    cloud_3pm = float(request.form['Cloud3pm'])
    humidity_9am = float(request.form['Humidity9am'])
    cloud_9am = float(request.form['Cloud9am'])
    wind_gust_speed = float(request.form['WindGustSpeed'])
    wind_speed_9am = float(request.form['WindSpeed9am'])
    min_temp = float(request.form['MinTemp'])
    wind_speed_3pm = float(request.form['WindSpeed3pm'])
    wind_gust_dir = int(request.form['WindGustDir'])

    # Define feature names
    feature_names = ['Humidity3pm', 'Rainfall', 'Cloud3pm', 'Humidity9am',
'Cloud9am', 'WindGustSpeed', 'WindSpeed9am', 'MinTemp', 'WindSpeed3pm',
'WindGustDir']

    # Create the data array in the specified order
    data = [humidity_3pm, rainfall, cloud_3pm, humidity_9am, cloud_9am,
wind_gust_speed, wind_speed_9am, min_temp, wind_speed_3pm, wind_gust_dir]
```

```
 # Create a DataFrame with the feature names
        input_data = pd.DataFrame([data], columns=feature_names)
        #print("Input Data:\n", input_data)

        input_data_scaled = scaler.transform(input_data)
        #print("Scaled Input Data:\n", input_data_scaled)
        input_data_scaled_df = pd.DataFrame(input_data_scaled,
columns=feature_names)

        # Make prediction
        prediction = model.predict(input_data_scaled_df)
        #print("Prediction:", prediction)

        # Assuming the model returns 1 for rain and 0 for no rain
        if prediction[0] == 1:
            return render_template('rain.html')
        else:
            return render_template('norain.html')

    return app
```

```
if __name__ == "__main__":

    app = create_app()

    app.run(debug=True)
```

Index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Rainfall prediction App</title>

    <link rel="stylesheet" href="../static/style.css">

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"

    rel="stylesheet"

    integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+AL
EwIH"

    crossorigin="anonymous">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.2/css/all.min.css" integrity="sha512-
SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3p
BnMFcV7oQPJkl9QevSCWr3W6A==" crossorigin="anonymous"
referrerpolicy="no-referrer" />

</head>


<body>
```

```
<nav class="navbar  bg-body-light border-bottom sticky-top">


    <div class="title container-fluid"> <i class="fa-solid fa-cloud-sun-rain
icon"></i> <h3 style="color: white;" class="mx-auto">Rainfall Prediction
app</h3> </div>


</nav>

<div class="row mt-3">

    <div class="col-8 offset-2">

    <br><br>

    <h3>Fill the details according to your location</h3>

        <form action="/predict" method="post">

        <i class="fa-solid fa-droplet text-success"></i>

        <div class="mb-3">

            <label for="Humidity3pm" class="form-label fw-bold text-
warning">Humidity at 3pm</label>

            <input name="Humidity3pm" placeholder="Add the Humidity,(Humidity
around 3pm is preferred)" type="text" class="form-control" required>

            <div class="valid-feedback">Title looks good</div>

        </div>


        <i class="fa-solid fa-cloud-rain text-success"></i>

        <div class="mb-3">

            <label for="Rainfall" class="form-label fw-bold text-
warning">Rainfall</label>

            <input name="Rainfall" placeholder="Add the amount of Rainfall
recorded" type="text" class="form-control" required>

            <div class="valid-feedback">Title looks good</div>

        </div>
```

```
<i class="fa-solid fa-cloud text-success"></i>

<div class="mb-3">

    <label for="Cloud3pm" class="form-label fw-bold text-warning">Clouds
at 3pm</label>

    <input name="Cloud3pm" placeholder="Add information about clouds
(around 3pm would be preferred)" type="text" class="form-control" required>

    <div class="valid-feedback">Title looks good</div>

</div>


<i class="fa-solid fa-droplet text-success"></i>

<div class="mb-3">

    <label for="Humidity9am" class="form-label fw-bold text-
warning">Humidity at 9am</label>

    <input name="Humidity9am" placeholder="Add the Humidity,(Humidity
around 9am)" type="text" class="form-control" required>

    <div class="valid-feedback">Title looks good</div>

</div>


<i class="fa-solid fa-cloud text-success"></i>

<div class="mb-3">

    <label for="Cloud9am" class="form-label fw-bold text-warning">Clouds
at 9am</label>

    <input name="Cloud9am" placeholder="Add information about clouds"
type="text" class="form-control" required>

    <div class="valid-feedback">Title looks good</div>

</div>


<i class="fa-solid fa-wind text-success"></i>

<div class="mb-3">
```

```
    <label for="WindGustSpeed" class="form-label fw-bold text-
warning">Wind Gust Speed</label>

    <input name="WindGustSpeed" placeholder="Add Wind Gust Speed"
type="text" class="form-control" required>

    <div class="valid-feedback">Title looks good</div>

  </div>



  <i class="fa-solid fa-wind text-success"></i>

  <div class="mb-3">

    <label for="WindSpeed9am" class="form-label fw-bold text-
warning">Wind Speed at 9am</label>

    <input name="WindSpeed9am" placeholder="Add WindSpeed at 9am"
type="text" class="form-control" required>

    <div class="valid-feedback">Title looks good</div>

  </div>


  <i class="fa-solid fa-temperature-low text-success"></i>

  <div class="mb-3">

    <label for="MinTemp" class="form-label fw-bold text-
warning">Minimum Temperature</label>

    <input name="MinTemp" placeholder="Add Minimum Temperature of
the day" type="text" class="form-control" required>

    <div class="valid-feedback">Title looks good</div>

  </div>


  <i class="fa-solid fa-wind text-success"></i>

  <div class="mb-3">
```

```
<label for="WindSpeed3pm" class="form-label fw-bold text-warning">Wind Speed at 3pm</label>

        <input name="WindSpeed3pm" placeholder="Add Wind Speed at 3pm" type="text" class="form-control" required>

        <div class="valid-feedback">Title looks good</div>

    </div>


    <i class="fa-solid fa-wind text-success"></i>

    <div class="mb-3">

        <label for="WindGustDir" class="form-label fw-bold text-warning">Wind Gust Direction</label>

        <select name="WindGustDir">

          <option value="0">E</option>

          <option value="1">ENE</option>

          <option value="2">ESE</option>

          <option value="3">N</option>

          <option value="4">NE</option>

          <option value="5">NNE</option>

          <option value="6">NNW</option>

          <option value="7">NW</option>

          <option value="8">S</option>

          <option value="9">SE</option>

          <option value="10">SSE</option>

          <option value="11">SSW</option>

          <option value="12">SW</option>

          <option value="13">W</option>

          <option value="14">WNW</option>

          <option value="15">WSW</option>
```

```
</select>

              <div class="valid-feedback">Title looks good</div>

            </div>




            <button class="btn btn-danger add-btn mt-3 fw-bold text-
white">Predict</button>

            </form>

        </div>

        </div>


        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIe
Hz" crossorigin="anonymous"></script>


</body>

</html>
```

Rain.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Rain Prediction</title>
    <style>
        body {
            background-image: url("https://images.unsplash.com/photo-
1697518725445-
037f24d787b4?w=500&auto=format&fit=crop&q=60&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8N3x8cmFpbnklMjB3ZWF0
aGVyfGVufDB8fDB8fHww");
```

```
            background-size: cover;
            color: white;
            text-align: center;
            padding-top: 20%;
            font-family: 'Arial', sans-serif;
        }
        h1 {
            font-size: 3em;
            margin-bottom: 0.5em;
            text-shadow: 0 0 10px rgba(255, 255, 255, 0.8), 0 0 20px rgba(255,
255, 255, 0.6), 0 0 30px rgba(255, 255, 255, 0.4);
            color: rgb(232, 128, 10);
        }
        p {
            font-size: 1.5em;
            margin-top: 0;
            text-shadow: 0 0 10px rgba(255, 255, 255, 0.8), 0 0 20px rgba(255,
255, 255, 0.6), 0 0 30px rgba(255, 255, 255, 0.4);
            color: rgb(232, 128, 10);
        }
    </style>
</head>
<body>
    <h1>Take an Umbrella!</h1>
    <p>It looks like it will rain tomorrow. Don't forget to take an umbrella
with you.</p>
</body>
</html>
```

No Rain.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>No Rain Prediction</title>
    <style>
        body {
```

```
          background-image: url("https://images.unsplash.com/photo-
1504370805625-d32c54b16100?w=500&auto=format&fit=crop&q=60&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8MTF8fGhvdCUyMHdlYXRoZX
J8ZW58MHx8MHx8fDA%3D");
          background-size: cover;
          color: white;
          text-align: center;
          padding-top: 20%;
          font-family: 'Arial', sans-serif;
        }
        h1 {
          font-size: 3em;
          margin-bottom: 0.5em;
          text-shadow: 0 0 10px rgba(255, 255, 255, 0.8), 0 0 20px rgba(255, 255,
255, 0.6), 0 0 30px rgba(255, 255, 255, 0.4);
          color: black;
        }
        p {
          font-size: 1.5em;
          margin-top: 0;
          text-shadow: 0 0 10px rgba(255, 255, 255, 0.8), 0 0 20px rgba(255, 255,
255, 0.6), 0 0 30px rgba(255, 255, 255, 0.4);
          color: black;
        }
    </style>
</head>
<body>
    <h1>Hurray! No Rain Tomorrow</h1>
    <p>No rain is expected tomorrow. You can go out and enjoy your day!</p>
</body>
</htm>
```

Style.css

```
body{
   background-image: url("https://images.unsplash.com/photo-1697518725445-037f24d787b4?w=500&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8N3x8cmFpbnklMjB3ZWF0aGVyfGVufDB8fDB8fHww");
   background-repeat: no-repeat;
   background-size: cover;
}
.navbar {
   height: 5rem;
   background-color: orange;
}
.title{
   display: flex;
   justify-content: center;
   align-items: center;
   min-width: 100vh;
}
.icon{
   color: rgb(16, 123, 211);
   font-size: 2rem;


}
```

10.2 GitHub & Project Demo Link

GitHub Link : https://github.com/ShaikMohammadHuzefa4531/Rainfall-Prediction-Using-Machine-Learning

Project Demo Link : https://drive.google.com/file/d/1SHNzso1C-Xifs8DdfpA-i0pT7a2JGYC-/view?usp=sharing