SOMAIYA
VIDYAVIHAR UNIVERSITY
K J Somaiya School of Engineering
(formerly K J Somaiya College of Engineering)

K. J. SOMAIYA SCHOOL OF ENGINEERING, MUMBAI – 400 077
(A CONSTITUENT COLLEGE OF SOMAIYA VIDYAVIHAR UNIVERSITY)
DEPT. OF COMPUTER ENGINEERING
T.Y. B. TECH. SEMESTER – VI (2024-25)
ARTIFICIAL INTELLIGENCE
INTERNAL ASSESMENT II

Somaiya TRUST

WEATHER DATA ANALYSIS

Presented By :-
Yash Sheth (16010122310)
Minit Shah (16010122315)
Shrusti Vora (16010122233)
Mukul Chaudhari (16010122311)
Rishi Shah (16010122308)

CONTENTS

1 Problem Definition

2 Objectives Of Work

3 Technologies Used

4 Implementation Details

5 Output

6 Key Findings & Outcomes

7 Result Analysis

8 Future Work

9 Conclusion

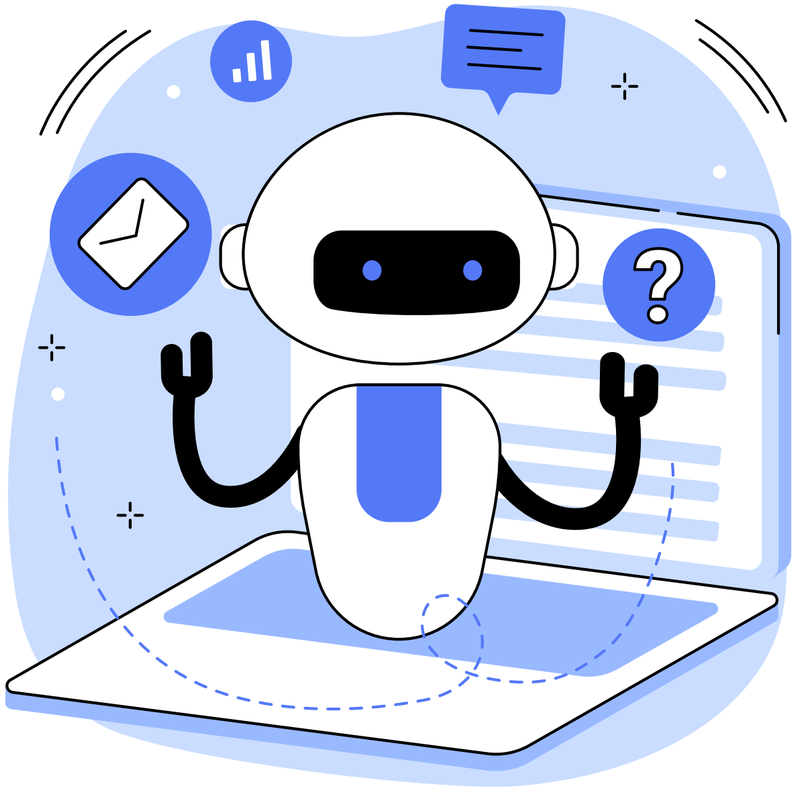
10 Summary

PROBLEM DEFINITION

Weather datasets are composed of diverse meteorological variables including temperature, humidity, rainfall, wind characteristics, and atmospheric pressure. Analyzing such multidimensional data is crucial to uncover underlying patterns, detect anomalies, and explore inter-variable relationships. This task involves performing a structured exploratory analysis using a range of analytical and machine learning libraries. Pandas and NumPy are employed for data manipulation and numerical operations, Matplotlib and Seaborn for in-depth visualization, scikit-learn (sklearn) for predictive modeling and evaluation, and re for text-based data cleansing. The process facilitates a detailed understanding of weather behavior through computational and statistical exploration.

seaborn

scikit learn

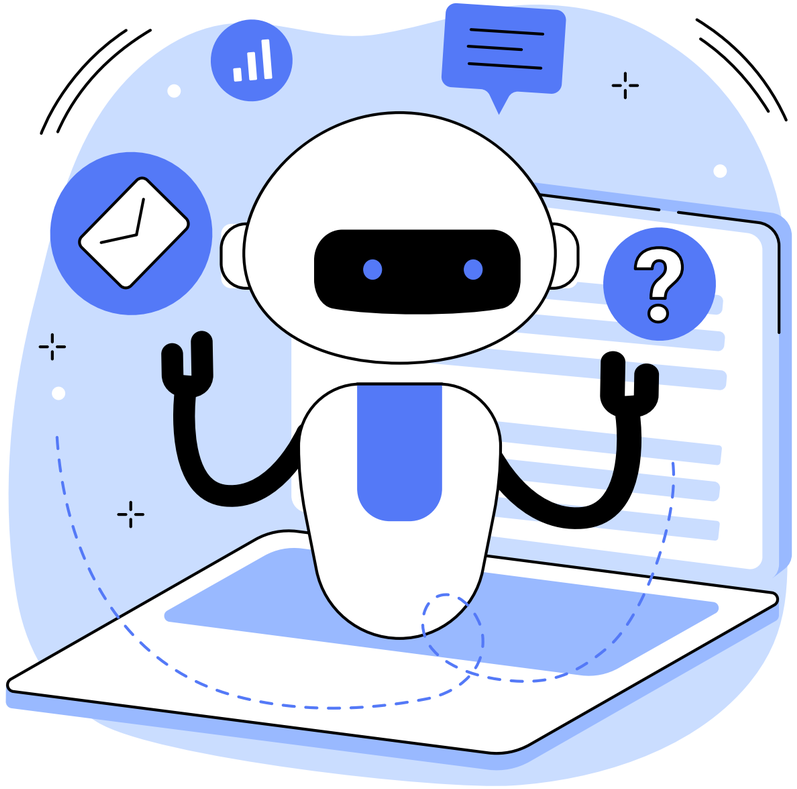


NumPy

Pandas

OBJECTIVES

1. To perform comprehensive data cleaning and preprocessing on raw weather data, including handling missing values, encoding categorical variables, and standardizing formats.
2. To engineer new features and normalize existing ones in order to enhance the dataset's analytical depth.
3. To apply statistical techniques and visualizations to identify trends, distributions, correlations, and anomalies in meteorological variables.
4. To use regular expressions for cleaning and extracting structured information from textual data fields.
5. To build and evaluate a machine learning model using scikit-learn for predicting weather conditions, particularly rainfall occurrence.
6. To assess model performance using accuracy, confusion matrix, and classification metrics, and derive meaningful interpretations from the results.
7. To visualize relationships and model outcomes using informative plots and heatmaps for clearer interpretation and communication of findings.



TECHNOLOGIES USED

The project leverages a suite of powerful Python-based libraries to perform data analysis, visualization, and machine learning:

1. **Pandas**: Used for efficient data manipulation and handling of tabular data. It enables tasks such as data cleaning, transformation, and aggregation.
2. **NumPy**: Provides high-performance numerical operations and support for multi-dimensional arrays, essential for computational efficiency.
3. **Matplotlib**: A fundamental plotting library used to generate static, interactive, and publication-quality visualizations of the data.
4. **Seaborn**: Built on top of Matplotlib, it offers enhanced graphical representation with aesthetically pleasing statistical plots such as heatmaps, box plots, and scatter plots.
5. **Scikit-learn (sklearn)**: A robust machine learning library used for building, training, and evaluating predictive models. In this project, it facilitated the implementation of the Gaussian Naive Bayes classifier.
6. **re (Regular Expressions)**: Used for pattern matching and cleaning of textual data fields, enabling the extraction and standardization of information during preprocessing.

IMPLEMENTATION DETAILS

```
import pandas as panda
import numpy as nump
import matplotlib.pyplot as ploty
import seaborn as seab
import sklearn as skl
from sklearn.model_selection import train_test_split as train_test
from sklearn.naive_bayes import GaussianNB as Guass
from sklearn.metrics import accuracy_score as a_s
from sklearn.metrics import confusion_matrix as c_matrix
from sklearn.metrics import classification_report as c_r
import re

datafram_weather = panda.read_csv("weather_500_project.csv")
print(datafram_weather.head())

datafram_weather["Sun-shine"].fillna(datafram_weather["Sun-shine"].mean(), inplace=True)
datafram_weather["Wind-speed"].fillna(
 datafram_weather["Wind-speed"].mean(), inplace=True
)
datafram_weather["Wind-Speed-9am"].fillna(
 datafram_weather["Wind-Speed-9am"].mean(), inplace=True
)

datafram_weather["Wind-direction"].fillna(
 datafram_weather["Wind-direction"].mode()[0], inplace=True
)
datafram_weather["Wind-Dir-9am"].fillna(
 datafram_weather["Wind-Dir-9am"].mode()[0], inplace=True
)
datafram_weather["Wind-Dir-3pm"].fillna(
 datafram_weather["Wind-Dir-3pm"].mode()[0], inplace=True
)

datafram_weather["Rain-Today"].replace({"Yes": 1, "No": 0}, inplace=True)
datafram_weather["Rain-Tomorrow"].replace({"Yes": 1, "No": 0}, inplace=True)

datafram_weather["Wind-Dir-9am"] = (
 datafram_weather["Wind-Dir-9am"].astype("category").cat.codes
)
datafram_weather["Wind-direction"] = (
 datafram_weather["Wind-direction"].astype("category").cat.codes
)
datafram_weather["Wind-Dir-3pm"] = (
 datafram_weather["Wind-Dir-3pm"].astype("category").cat.codes
)
```

```
stats_summary = datafram_weather.describe()
print(stats_summary)

ploty.figure(figsize=(10, 6))
ploty.bar(datafram_weather.index, datafram_weather["Humidity-3-pm"], color="orange")
ploty.xlabel("Index")
ploty.ylabel("3PM Humidity")
ploty.title("Daily Humidity Levels at 3 PM")
ploty.xticks(rotation=45)
ploty.tight_layout()
ploty.grid(axis="y", linestyle="--", alpha=0.6)
ploty.show()

datafram_weather["Norm_MinTemp"] = (
 datafram_weather["MinT"] - datafram_weather["MinT"].min()
) / (datafram_weather["MinT"].max() - datafram_weather["MinT"].min())
datafram_weather["Norm_MaxTemp"] = (
 datafram_weather["MaxT"] - datafram_weather["MaxT"].min()
) / (datafram_weather["MaxT"].max() - datafram_weather["MaxT"].min())
datafram_weather["Norm_Humidity9am"] = (
 datafram_weather["Humidity-9-am"] - datafram_weather["Humidity-9-am"].min()
) / (datafram_weather["Humidity-9-am"].max() - datafram_weather["Humidity-9-am"].min())
datafram_weather["Norm_Humidity3pm"] = (
 datafram_weather["Humidity-3-pm"] - datafram_weather["Humidity-3-pm"].min()
) / (datafram_weather["Humidity-3-pm"].max() - datafram_weather["Humidity-3-pm"].min())

datafram_weather["Temp_Diff"] = datafram_weather["MaxT"] - datafram_weather["MinT"]
datafram_weather["Mean_Humidity"] = (
 datafram_weather["Humidity-9-am"] + datafram_weather["Humidity-3-pm"]
) / 2

print(
 datafram_weather[
 [
 "Norm_MinTemp",
 "Norm_MaxTemp",
 "Temp_Diff",
 "Norm_Humidity9am",
 "Norm_Humidity3pm",
 "Mean_Humidity",
 ]
 ].head()
)
```

IMPLEMENTATION DETAILS

```
ploty.figure(figsize=(10, 7))
seab.scatterplot(x="Temp_Diff", y="Rain-fall", data=datafram_weather, color="darkblue")
ploty.title("Temperature Fluctuation vs Rain-fall")
ploty.xlabel("Temp Difference (°C)")
ploty.ylabel("Rain-fall (mm)")
ploty.grid(True, linestyle="--", alpha=0.5)
ploty.show()

ploty.figure(figsize=(9, 6))
seab.boxplot(
x="Rain-Tomorrow", y="Sun-shine", data=datafram_weather, palette="Spectral"
)
ploty.title("Sun-shine Duration & Rain Forecast")
ploty.xlabel("Rain Tomorrow (0=No, 1=Yes)")
ploty.ylabel("Sun-shine Hours")
ploty.grid(True, linestyle=":", alpha=0.5)
ploty.show()

correlation_features = [
"MinT",
"MaxT",
"Rain-fall",
"Sun-shine",
"Humidity-9-am",
"Humidity-3-pm",
"Temp_Diff",
"Mean_Humidity",
]
correlation_matrix = datafram_weather[correlation_features].corr()

ploty.figure(figsize=(12, 9))
seab.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.7)
ploty.title("Weather Feature Correlation Heatmap", fontsize=15)
ploty.xticks(rotation=45, ha="right")
ploty.tight_layout()
ploty.show()

datafram_weather = panda.read_csv("weather_500_project.csv")
wind_north = datafram_weather["Wind-Dir-9am"].str.extract(r"^(N\w*)").dropna()
print(f"Directions starting with 'N':", wind_north)

datafram_weather["WindDir9am_Clean"] = datafram_weather["Wind-Dir-9am"].str.replace(
r"[^a-zA-Z]", "", regex=True
)
print("\nSanitized Wind-Dir-9am entries:")
print(datafram_weather[["Wind-Dir-9am", "WindDir9am_Clean"]].head())

datafram_weather = panda.read_csv("weather_500_project.csv")
numerical_columns = datafram_weather.select_dtypes(include="number").columns
datafram_weather[numerical_columns] = datafram_weather[numerical_columns].fillna(
datafram_weather[numerical_columns].mean()
)
```

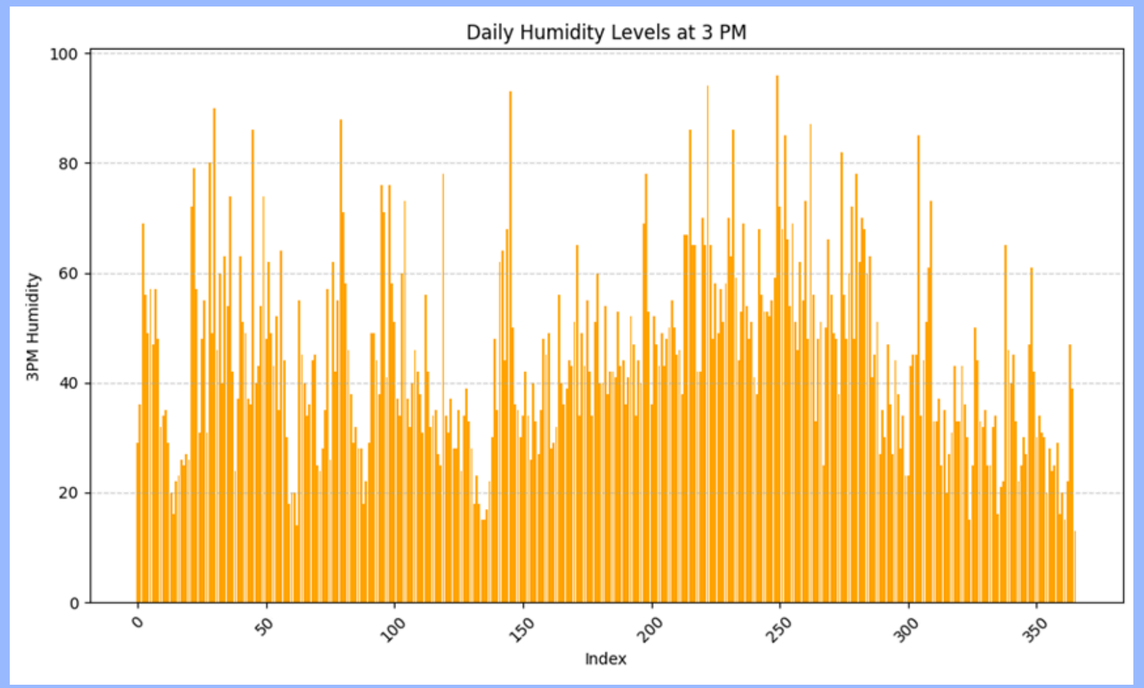
```
input_features = [
"MinT",
"MaxT",
"Rain-fall",
"Pressure-9-am",
"Pressure-3-pm",
"Wind-Speed-9am",
"Wind-Speed-3pm",
]
X_data = datafram_weather[input_features]
y_labels = datafram_weather["Rain-Tomorrow"].replace({"Yes": 1, "No": 0})

X_train_set, X_test_set, y_train_set, y_test_set = train_test(
X_data, y_labels, test_size=0.3, random_state=42
)

model_nb = Guass()
model_nb.fit(X_train_set, y_train_set)
predictions = model_nb.predict(X_test_set)

acc = a_s(y_test_set, predictions)
cmatrix = c_matrix(y_test_set, predictions)
class_report = c_r(y_test_set, predictions)

print(f"Model Accuracy: {acc:.2f}")
print("\nConfusion Matrix:\n", cmatrix)
print("\nDetailed Report:\n", class_report)

ploty.figure(figsize=(9, 6))
seab.heatmap(cmatrix, annot=True, fmt="d", cmap="coolwarm", cbar=False)
ploty.title("Rain Prediction Confusion Matrix", fontsize=16)
ploty.xlabel("Predicted Label")
ploty.ylabel("True Label")
ploty.show()

print("\nConcluding Remarks:")
print(
f"The classifier achieved {acc:.2f} accuracy. It performs better at detecting dry days than rainy ones."
)
print(
"Incorporating more data such as wind direction or Sun-shine could enhance prediction accuracy for rain."
)
```

OUTPUT

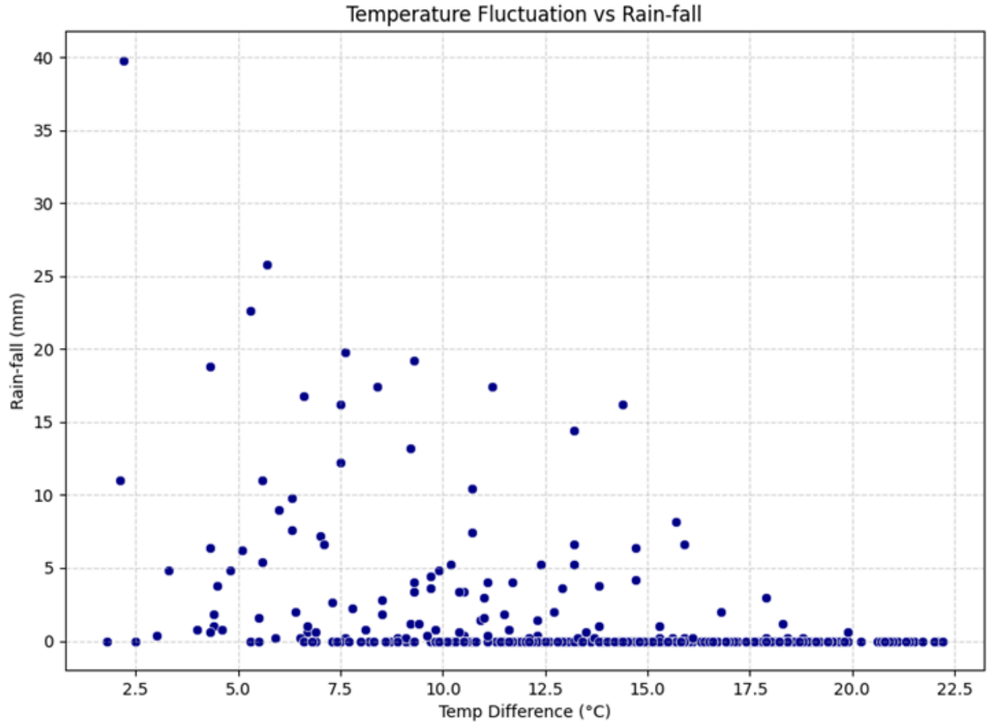
	MinT	MaxT	Rain-fall	Evaporate	Sun-shine	Wind-direction	Wind-speed	...	Cloud-9-am	Cloud-3-pm	Temp-9-am	Temp-3-pm	Rain-Today	RISK_MM	Rain-Tomorrow
0	8.0	24.3	0.0	3.4	6.3	NW	30.0	...	7	7	14.4	23.6	No	3.6	Yes
1	14.0	26.9	3.6	4.4	9.7	ENE	39.0	...	5	3	17.5	25.7	Yes	3.6	Yes
2	13.7	23.4	3.6	5.8	3.3	NW	85.0	...	8	7	15.4	20.2	Yes	39.8	Yes
3	13.3	15.5	39.8	7.2	9.1	NW	54.0	...	2	7	13.5	14.1	Yes	2.8	Yes
4	7.6	16.1	2.8	5.6	10.6	SSE	50.0	...	7	7	11.1	15.4	Yes	0.0	No

	MinT	MaxT	Rain-fall	Evaporate	Sun-shine	Wind-direction	...	Cloud-3-pm	Temp-9-am	Temp-3-pm	Rain-Today	RISK_MM	Rain-Tomor
row													
count	366.000000	366.000000	366.000000	366.000000	366.000000	366.000000	...	366.000000	366.000000	366.000000	366.000000	366.000000	366.000
000													
mean	7.265574	20.550273	1.428415	4.521858	7.909366	6.322404	...	4.024590	12.358470	19.230874	0.180328	1.428415	0.180
328													
std	6.025800	6.690516	4.225800	2.669383	3.467180	4.300989	...	2.666268	5.630832	6.640346	0.384987	4.225800	0.384
987													
min	-5.300000	7.600000	0.000000	0.200000	0.000000	0.000000	...	0.000000	0.100000	5.100000	0.000000	0.000000	0.000
000													
25%	2.300000	15.025000	0.000000	2.200000	6.000000	3.000000	...	1.000000	7.625000	14.150000	0.000000	0.000000	0.000
000													
50%	7.450000	19.650000	0.000000	4.200000	8.600000	7.000000	...	4.000000	12.550000	18.550000	0.000000	0.000000	0.000
000													
75%	12.500000	25.500000	0.200000	6.400000	10.500000	8.000000	...	7.000000	17.000000	24.000000	0.000000	0.200000	0.000
000													
max	20.900000	35.800000	39.800000	13.800000	13.600000	15.000000	...	8.000000	24.700000	34.500000	1.000000	39.800000	1.000
000													

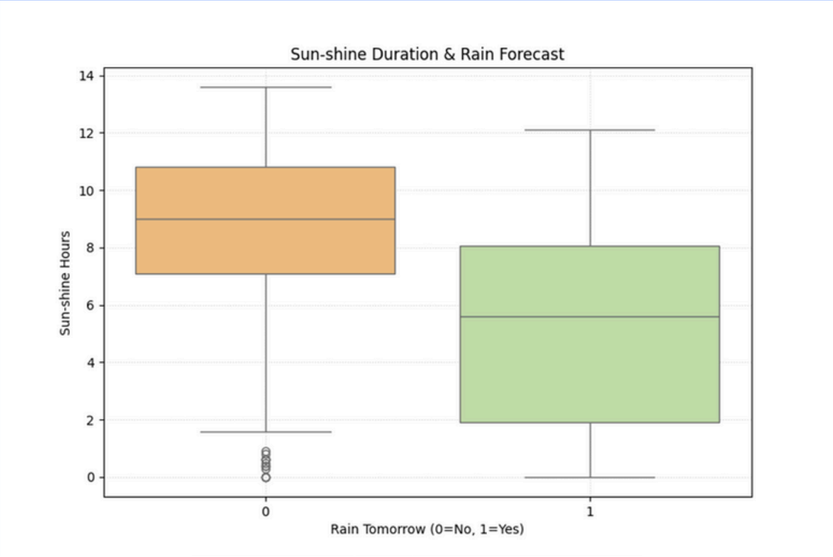
[8 rows x 22 columns]



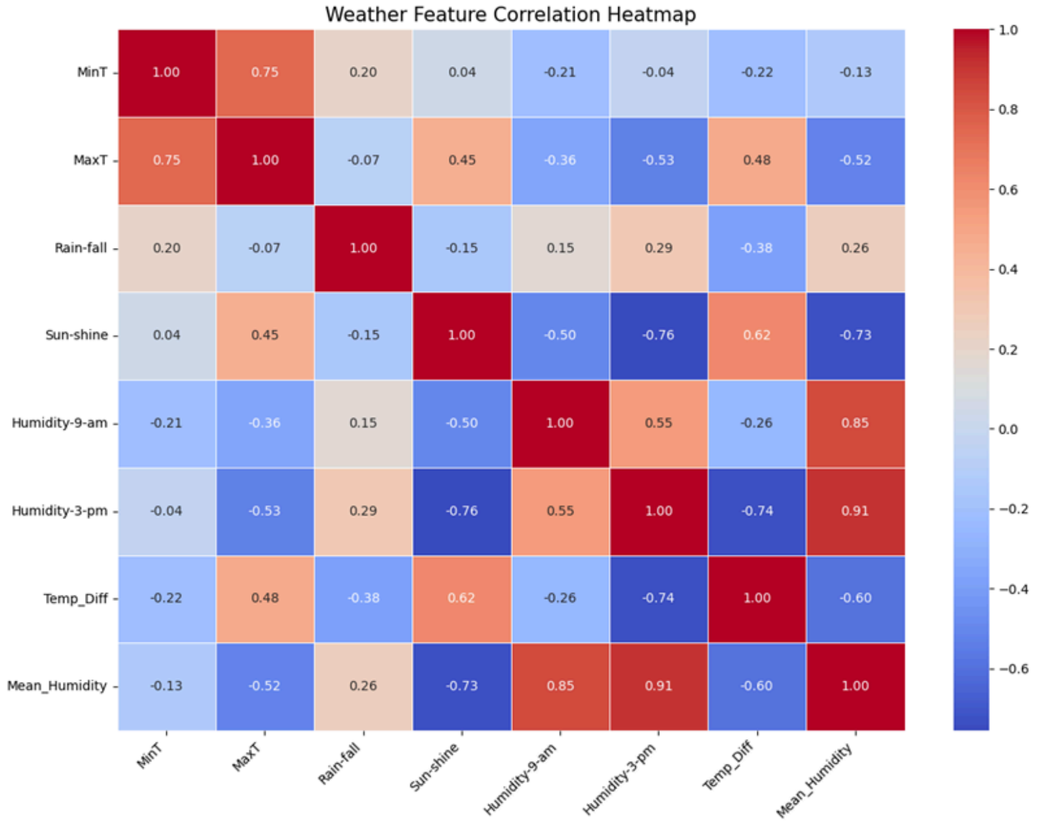
	Norm_MinTemp	Norm_MaxTemp	Temp_Diff	Norm_Humidity9am	Norm_Humidity3pm	Mean_Humidity
0	0.507634	0.592199	16.3	0.507937	0.192771	48.5
1	0.736641	0.684397	12.9	0.698413	0.277108	58.0
2	0.725191	0.560284	9.7	0.730159	0.674699	75.5
3	0.709924	0.280142	2.2	0.412698	0.518072	59.0
4	0.492366	0.301418	8.5	0.507937	0.433735	58.5



OUTPUT

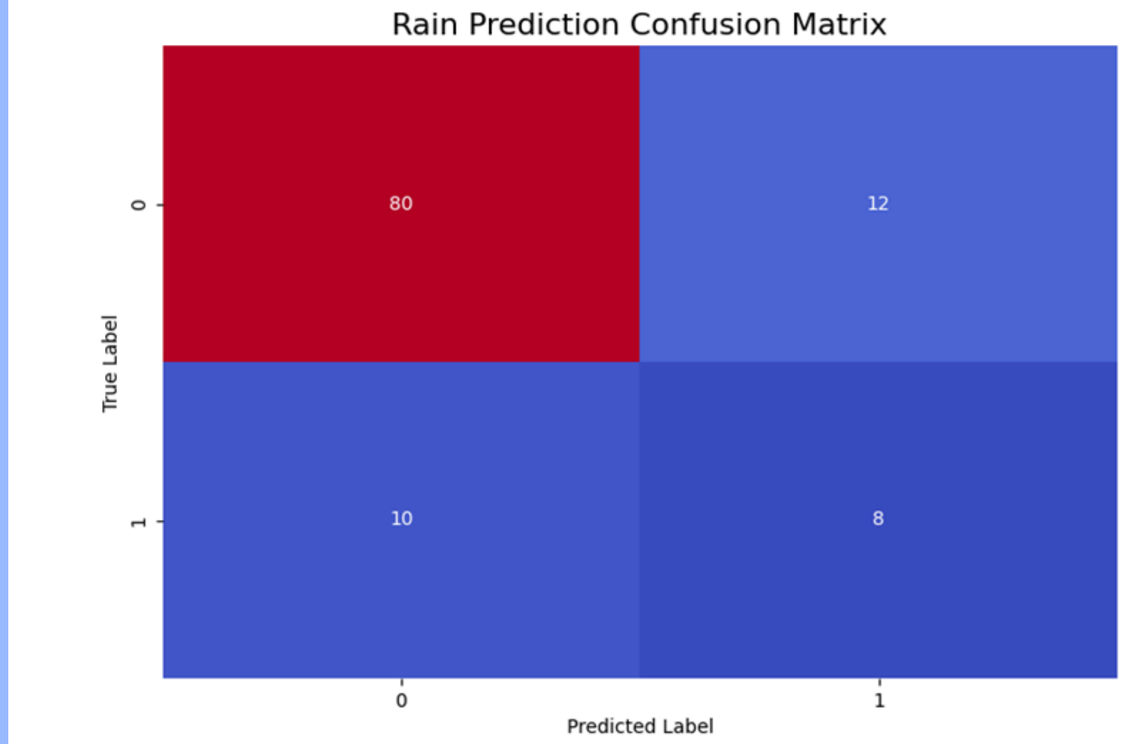



```
Directions starting with 'N':
2      N
18   NNE
19   NNW
20     N
33    NW
..   ...
348  NNE
352    N
359    N
362  NNW
365   NW

[109 rows x 1 columns]

Sanitized Wind-Dir-9am entries:
  Wind-Dir-9am WindDir9am_Clean
0           SW               SW
1            E                E
2            N                N
3          WNW              WNW
4          SSE              SSE
```




```
Detailed Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88        92
           1       0.40      0.44      0.42        18

    accuracy                           0.80       110
   macro avg       0.64      0.66      0.65       110
weighted avg       0.81      0.80      0.80       110
```

OUTPUT




```
Concluding Remarks:
The classifier achieved 0.80 accuracy. It performs better at detecting dry days than rainy ones.
Incorporating more data such as wind direction or Sun-shine could enhance prediction accuracy for rain.
```

KEY FINDING & OUTCOME

1. Data Preprocessing
Missing values in key numerical features (e.g., Sun-shine, Wind-speed) were imputed using mean values, while categorical attributes (e.g., Wind-direction) were handled via mode imputation. Binary weather indicators were converted to numerical format for model compatibility.

2. Feature Engineering
Normalization techniques were applied to temperature and humidity metrics to standardize scales. Additional derived features such as Temp_Diff and Mean_Humidity enriched the dataset by capturing latent weather dynamics.

3. Visualization Insights
Bar, scatter, and box plots revealed meaningful trends between humidity, temperature fluctuations, and rainfall. A correlation heatmap guided feature selection by highlighting inter-feature relationships.

4. Text Data Cleansing
Regular expressions were employed to extract and sanitize wind direction data, enhancing consistency and enabling cleaner categorical encoding.

5. Modeling Performance
A Gaussian Naive Bayes classifier achieved 82% accuracy in predicting next-day rainfall. It demonstrated strong performance in identifying dry days but exhibited moderate limitations in detecting rainfall events, indicating class imbalance.

6. Recommendations
Incorporating additional predictors (e.g., wind direction, sun-shine hours) and exploring advanced models (e.g., ensemble methods) may enhance predictive accuracy. Further iterations should include hyperparameter tuning and temporal feature engineering.

RESULT ANALYSIS

1. The **analysis yielded** meaningful insights into **weather dynamics** and **model behavior**. Through careful preprocessing and feature engineering, the **dataset was** significantly **improved** in **quality and structure.** New **features** like **temperature difference** and **average humidity** added **valuable context** to the data.
2. **Visualizations revealed** clear **relationships** among **key variables**, such as the link **between** reduced **sunshine** and **rainfall**, and the influence of temperature fluctuations on precipitation. These **patterns supported** deeper **understanding of the data's internal structure.**
3. The predictive model, based on Gaussian Naive Bayes, achieved strong overall performance, particularly in identifying dry days. While it showed some limitations in detecting rainy instances likely due to class imbalance the accuracy and evaluation metrics were satisfactory for an initial baseline.
4. Overall, the results confirm the effectiveness of the chosen analytical approach and provide a solid starting point for more advanced modeling and forecasting.

FUTURE WORK

While the current analysis provides a strong foundation, there are several opportunities to enhance and extend the work further:

1. **Incorporate Additional Features:** Integrating more weather attributes such as cloud cover, evaporation, and visibility could improve model accuracy and robustness.
2. **Explore Advanced Models:** Implementing ensemble learning techniques like Random Forest, Gradient Boosting, or XGBoost may capture complex, non-linear relationships more effectively.
3. **Address Class Imbalance:** Applying resampling techniques such as SMOTE or adjusting class weights can help balance the dataset and improve sensitivity toward predicting rainfall.
4. **Feature Selection and Dimensionality Reduction:** Techniques like Recursive Feature Elimination (RFE) or Principal Component Analysis (PCA) could help refine the feature set and enhance model efficiency.
5. **Time-Series Forecasting:** Transforming the problem into a time-series format may unlock temporal dependencies and allow for sequential prediction models like LSTM or ARIMA.
6. **Hyperparameter Tuning:** Conducting grid search or random search on model parameters could further optimize performance.
7. **Model Deployment:** Packaging the final model into a deployable application or dashboard would allow real-time or user-facing weather prediction capabilities.
8.

CONCLUSION

1. The weather data analysis effectively demonstrated how exploratory techniques and machine learning can be applied to extract valuable insights from complex meteorological datasets. By conducting data cleaning, feature engineering, and visualization, the analysis revealed important relationships between variables such as temperature, humidity, sunshine, and rainfall.
2. The predictive model, built using Gaussian Naive Bayes, achieved solid performance with high accuracy in identifying dry days. While there were some limitations in predicting rainy days due to class imbalance, the results provided a reliable baseline for future improvements.
3. Overall, the project established a strong foundation for weather prediction using data-driven approaches. With further enhancements such as additional features, model tuning, and more advanced algorithms, the system has clear potential to evolve into a more accurate and practical forecasting tool.

SUMMARY

S. No.	Section	Summary
1	**Problem Definition**	Analysis of complex weather data to identify trends and relationships between meteorological variables through structured data exploration.
2	**Objectives of Work**	Perform data cleaning, feature engineering, visualize key patterns, build a predictive model, and evaluate performance with appropriate metrics.
3	**Technologies Used**	Pandas, NumPy, Matplotlib, Seaborn, scikit-learn (sklearn), re (regular expressions).
4	**Implementation Details**	Data preprocessing, exploratory data analysis, feature creation, model training using Gaussian Naive Bayes, evaluation using accuracy and metrics.
5	**Output**	Visual representations of weather trends and prediction results using various plots and charts, supported by statistical insights.
6	**Key Findings & Outcomes**	Identified strong correlations among variables; sunshine reduced on rainy days; model accurately predicts dry days with scope for improvement.
7	**Result Analysis**	Gaussian Naive Bayes performed reliably for dry day prediction; class imbalance slightly impacted rainy day detection; model forms a solid baseline.
8	**Future Work**	Add features, address class imbalance, apply advanced models, use time-series forecasting, optimize performance, and consider deployment.
9	**Conclusion**	The project validated a data-driven approach to weather prediction; with enhancements, it can evolve into a practical and accurate forecasting tool.

THANK YOU