RAINFALL PREDICTION

A research project report submitted in partial fulfillment.
of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (Internet of Things)

Submitted by

I. SAI PAVAN	**20BQ1A4918**
B. HARICHARAN REDDY	**20BQ1A4910**
K. NAVEEN CHAND	**21BQ5A4901**

under the esteemed guidance of

Mrs. M. Rajya Lakshmi

Assoc. Professor

[Program: Computer Science and Engineering (Internet of Things) – CSO]

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

(Autonomous)

Approved by AICTE, Permanently Affiliated to JNTUK, NAAC Accredited with 'A' Grade, ISO 9001:2015 Certified

Nambur (V), Pedakakani (M), Guntur (Dt.), Andhra Pradesh – 522 508

2023

[Program: Computer Science and Engineering (Internet of Things) – CSO]

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

(Autonomous)

Approved by AICTE, Permanently Affiliated to JNTUK, NAAC Accredited with 'A' Grade, ISO 9001:2015 Certified

Nambur (V), Pedakakani (M), Guntur (Dt.), Andhra Pradesh – 522 508

CERTIFICATE

This is to certify that the research project report entitled "**RAINFALL PREDICTION"** is being submitted by **I. Sai Pavan (**Regd.No**: 20BQ1A4910**), **B.Haricharan Reddy (**Regd.No**: 20BQ1A4918**), **K. Naveen Chand (**Regd.No**: 21B5A4901**) in partial fulfillment of the requirement for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering (Internet of Things)** to the Vasireddy Venkatadri Institute of Technology is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this project have not been submitted to any other university or institute for the award of any degree or diploma.

Signature of the Supervisor

Mrs. M. Rajya Lakshmi
Associate Professor,
Department of CSO, VVIT.

Head of the Department

Dr. Chintalapudi V Suresh
Professor & HoD,
Department of CSO, VVIT.

DECLARATION

We hereby declare that the work embodied in this research project entitled "**Rainfall Prediction**", which is being submitted by us in requirement for the B. Tech Degree in **Computer Science and Engineering (Internet of Things)** from Vasireddy Venkatadri Institute of Technology, is the result of investigations carried out by us under the supervision of Mrs. M. Rajya Lakshmi, Associate Professor.

The work is original and the results in this thesis have not been submitted elsewhere for the award of any degree or diploma.

Signature of the Candidates

I. SAI PAVAN (Regd.No**: 20BQ1A4918**)

B. HARICHARAN REDDY (Regd.No**: 20BQ1A4910**)

K. NAVEEN CHAND (Regd.No**: 21BQ5A4901**)

Department Vision

To accomplish the aspirations of emerging engineers to attain global intelligence by obtaining computing and design abilities through communication that elevate them to meet the needs of industry, economy, society, environmental and global.

Department Mission

- To mould the fresh minds into highly competent IoT application developers by enhancing their knowledge and skills in diverse hardware and software design aspects for covering technologies and multi-disciplinary engineering practices.
- To provide the sate- of- the art facilities to forge the students in industry-ready in IoT system development.
- To nurture the sense of creativity and innovation to adopt the socio-economic related activities.
- To promote collaboration with the institutes of national and international repute with a view to have best careers.
- To enable graduates to emerge as independent entrepreneurs and future leaders.

Program Educational Objectives (PEOs)

PEO-1: To formulate the engineering practitioners to solve industry's technological problems.

PEO-2: To engage the engineering professionals in technology development, deployment, and engineering system implementation.

PEO-3: To instill professional ethics, values, social awareness, and responsibility to emerging technology leaders.

PEO-4: To facilitate interaction between students and peers in other disciplines of industry and society that contribute to the economic growth**.**

PEO-5: To provide the technocrats the amicable environment for the successful pursuing of engineering and management.

PEO-6: To create right path to pursue their careers in teaching, research, and innovation.

Program Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

PSO-1: Proficient and innovative with a strong cognizance in the arenas of sensors, IoT, data science, controllers and signal processing through the application of acquired knowledge and skills.

PSO-2: Apply cutting-edge techniques and tools of sensing and computation to solve multi-disciplinary challenges in industry and society.

PSO-3: Exhibit independent and collaborative research with strategic planning while demonstrating professional and ethical responsibilities of the engineering profession.

Project Outcomes

Students who complete a minor project will:

PW-01. Use the design principles and develop concept for the project

PW-02. Estimate the time frame and cost for the project execution and completion.

PW-03. Analyze the project progress with remedial measures individual in a team.

PW-04. Examine the environmental impact of the project.

PW-05. Demonstrate the project functionality along with report and presentation.

PW-06. Apply the Engineering knowledge in design and economically manufacturing of components to support the society need.

PW-07. Assess health, safety and legal relevant to professional engineering practices.

PW-08. Comply the environmental needs and sustainable development.

PW-09. Justify ethical principles in engineering practices.

PW-010. Perform multi-disciplinary task as an individual and / or team member to manage the project/task.

PW-011. Comprehend the Engineering activities with effective presentation and report.

PW-012. Interpret the findings with appropriate technological / research citation.

MAPPING OF PROJECT OUTCOMES TO POs

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
PW-01	3	2	2	2								
PW-02	3	2	2								3	
PW-03	3	3		2	3					3		
PW-04	3					3	3	3				3
PW-05	3	2									3	
PW-06	3	2	2	2	3							
PW-07						3						
PW-08							3					
PW-09								3				
PW-10									3		3	
PW-11										3		
PW-12												3
PW-PO	**3**	**2**	**2**	**2**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**

MAPPING OF PROJECT OUTCOMES TO PSOs

	PSO1	PSO2	PSO3
PW-01	2	2	2
PW-02			2
PW-03			
PW-04	3	3	3
PW-05			
PW-06	2	2	2
PW-07	2	2	2
PW-08	1	1	1
PW-09	2	2	2
PW-10	2	2	2
PW-11	2	2	2
PW-12	1	1	1
PW-PSO	**2**	**2**	**2**

Note: Strong – 3, Moderate – 2, Low – 1

CONTENTS

		Page No
ABSTRACT		x
LIST OF TABLES		xi
LIST OF FIGURES		xii
CHAPTER-1 INTRODUCTION		**01**
1.1	Background	02
1.2	Motivation	02
1.3	Objectives	02
1.4	Need to Study	03
CHAPTER-2 DATA PREPARATION TECHNIQUES		**04**
2.1	Importing the Data	05
2.2	Data Exploration	05
2.3	Handling Class Imbalance For Rainfall Prediction	07
2.4	Imputation and Transformation	08
2.5	Feature Selection for Rainfall Prediction	12
CHAPTER-3 MACHINE LEARNING APPROACHES		**13**
3.1	Training Rainfall Prediction Model with Different Models	14
3.1.1	Logistic Regression	14
3.1.2	Decision Tree	15
3.1.3	Neural Network	15
3.1.4	Random Forest	16
3.1.5	LightGBM	17
3.1.6	CatBoost	17
3.1.7	XGBoost	17
3.2	Plotting Decision Region for all Models	18
CHAPTER-4 RESULTS & ANALYSIS		**20**
4.1	Rainfall Prediction Model Comparison	21
CHAPTER-5 CONCLUSION AND FUTURE SCOPE		**23**
5.1	Conclusion	24
5.2	Future scope	24
5.2.1	Ensemble Models	24
5.2.2	Hyperparameter Tuning	24
5.2.3	Spatiotemporal Analysis	25
5.2.4	Streaming Data	25
5.2.5	Hybrid Models	25

5.2.6	User-Friendly Interfaces	25
REFERENCES		**26**
APPENDIX		**28**

ABSTRACT

Rainfall prediction is a critical aspect of weather forecasting and water resource management. This project focuses on the development and comparison of machine learning models for rainfall prediction. We employ various algorithms, including logistic regression, decision tree, neural network, random forest, LightGBM, CatBoost, and XGBoost, to train and evaluate rainfall prediction models. The dataset used in this study encompasses historical weather data, such as temperature, humidity, wind speed, and atmospheric pressure, collected over several years. By utilizing these features, each machine learning algorithm is trained to forecast rainfall events accurately. The project's primary objective is to assess the performance of these models in terms of accuracy, precision, recall, and F1-score. Furthermore, we employ various evaluation metrics, including confusion matrices and receiver operating characteristic (ROC) curves, to provide a comprehensive comparison of the models. Our findings reveal the strengths and weaknesses of each algorithm in rainfall prediction, shedding light on which models perform optimally under different weather conditions and geographic locations. This research contributes valuable insights into the application of machine learning techniques in enhancing rainfall prediction, ultimately aiding in better preparedness and mitigation of weather-related risks.

LIST OF TABLES

Table.No	Description	Page No
3.1	Performance Measures for Logistic Regression	15
3.2	Performance Measures for Decision Tree	15
3.3	Performance Measures for Neural Network	16
3.4	Performance Measures for Random Forest	16

LIST OF FIGURES

Figure No	Description	Page No
2.1	RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset	6
2.2	RainTomorrow Indicator No(0) and Yes(1) after Oversampling	7
2.3	Missing Data Pattern in Training Data	7
2.4	Correlation Heatmap	10
2.5	Pairwise correlation between highly correlated characteristics	11
3.1	Plotting Decision Region for all Models	19
4.1	Model Comparison: Accuracy and Time taken for execution.	21
4.2	Model Comparison: Area under ROC and Cohens Kappa	22

CHAPTER 1

INTRODUCTION

1.1 Background

Weather forecasting plays a pivotal role in our daily lives, influencing a wide range of activities from agriculture and infrastructure planning to disaster management. Among the various weather phenomena, rainfall is of particular significance due to its direct impact on water resources, agriculture, and the environment. Accurate rainfall prediction is vital for making informed decisions in these domains, and the advancement of machine learning has opened new avenues for enhancing our ability to forecast rainfall events.

1.2 Motivation

Traditional rainfall prediction methods rely on meteorological models and historical climate data. While these methods have proven effective to some extent, they often struggle to capture the complexity and nuances of rainfall patterns. Machine learning techniques, on the other hand, offer a data-driven approach that can uncover hidden patterns and relationships within large datasets, potentially leading to more accurate and reliable rainfall forecasts.

1.3 Objectives

The primary objective of this research project is to develop and compare rainfall prediction models using a variety of machine learning algorithms. Specifically, we aim to:

- Utilize logistic regression, decision tree, neural network, random forest, LightGBM, CatBoost, and XGBoost algorithms to train rainfall prediction models.
- Evaluate the performance of these models using a comprehensive set of metrics, including accuracy, precision, recall, and F1-score.
- Investigate the strengths and weaknesses of each algorithm in the context of rainfall prediction, considering different geographic locations and weather conditions.
- Provide valuable insights into the application of machine learning in improving rainfall forecasting accuracy, with implications for agriculture, water resource management, and disaster preparedness.

1.4 Need to Study

This research project addresses a critical need in the field of weather forecasting and environmental management. The existing methods for rainfall prediction, primarily relying on meteorological models and historical data, often fall short in providing precise and reliable forecasts. This deficiency can have far-reaching consequences, impacting agriculture, water resource management, disaster preparedness, and more.

The emergence of machine learning as a powerful tool for data-driven insights offers a promising solution to enhance rainfall prediction. However, there remains a need to comprehensively assess and compare various machine learning algorithms to determine their effectiveness in this specific domain. Understanding which algorithms excel under different conditions and geographic regions is crucial for advancing the accuracy of rainfall forecasts.

This research project aims to fill this gap by systematically evaluating and comparing the performance of seven distinct machine learning algorithms in rainfall prediction. The insights gained from this study can help improve our understanding of the applicability of these algorithms in practical scenarios and contribute to more accurate and timely rainfall forecasts, ultimately benefiting a wide range of sectors dependent on weather-related information.

CHAPTER 2
DATA PREPARATION TECHNIQUES

2.1 Importing the Data

Let's start this task of rainfall prediction by importing the data.

```
import pandas as pd
from google.colab import files
full_data = pd.read_csv('weatherAUS.csv')
full_data.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	Pressure3pm	Cloud9a
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	22.0	1007.7	1007.1	8.
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	25.0	1010.6	1007.8	Na
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	30.0	1007.6	1008.7	Na
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	16.0	1017.6	1012.8	Na
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	33.0	1010.8	1006.0	7.

5 rows × 24 columns

2.2 Data Exploration

We will first check the number of rows and columns. Next, we'll check the size of the dataset to decide if it needs size compression.

```
full_data.shape
```

```
(63535, 24)
```

```
full_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63535 entries, 0 to 63534
Data columns (total 24 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           63535 non-null  object
 1   Location       63535 non-null  object
 2   MinTemp        63108 non-null  float64
 3   MaxTemp        63324 non-null  float64
 4   Rainfall       62781 non-null  float64
 5   Evaporation    27553 non-null  float64
 6   Sunshine       21028 non-null  float64
 7   WindGustDir    58084 non-null  object
 8   WindGustSpeed  58091 non-null  float64
 9   WindDir9am     57202 non-null  object
 10  WindDir3pm     61053 non-null  object
 11  WindSpeed9am   62425 non-null  float64
 12  WindSpeed3pm   61817 non-null  float64
 13  Humidity9am    62491 non-null  float64
 14  Humidity3pm    62038 non-null  float64
 15  Pressure9am    53855 non-null  float64
 16  Pressure3pm    53903 non-null  float64
 17  Cloud9am       37046 non-null  float64
 18  Cloud3pm       36263 non-null  float64
 19  Temp9am        62814 non-null  float64
 20  Temp3pm        62305 non-null  float64
 21  RainToday      62780 non-null  object
 22  RISK_MM        63534 non-null  float64
 23  RainTomorrow   63534 non-null  object
dtypes: float64(17), object(7)
memory usage: 11.6+ MB
```

" RainToday" and "RainTomorrow" are objects (Yes / No). I will convert them to binary (1/0) for our convenience.

```
full_data['RainToday'].replace({'No': 0, 'Yes': 1},inplace = True)
full_data['RainTomorrow'].replace({'No': 0, 'Yes': 1},inplace = True)
```

Next, we will check if the dataset is unbalanced or balanced. If the data set is unbalanced, we need to either down sample the majority or oversample the minority to balance it.

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (8,5))
full_data.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['skyblue','navy'], alpha = 0.9, rot=0)
plt.title('RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset')
plt.show()
```

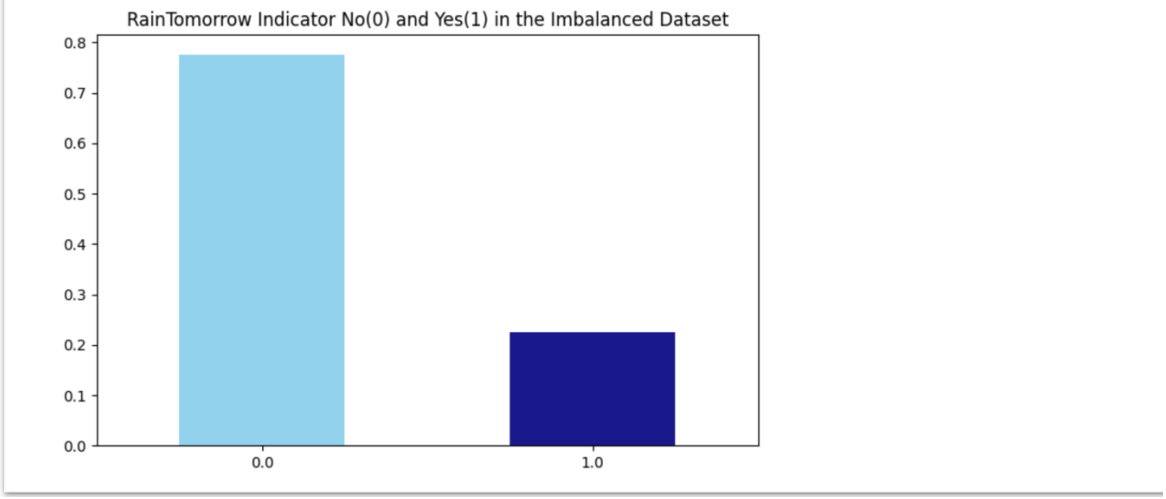

Fig 2.1 RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset

We can observe that the presence of "0" and "1" is almost in the 78:22 ratio. So, there is a class imbalance, and we have to deal with it. To fight against the class imbalance, we will use here the oversampling of the minority class. Since the size of the dataset is quite small, majority class subsampling wouldn't make much sense here.

2.3 Handling Class Imbalance for Rainfall Prediction

```
from sklearn.utils import resample

no = full_data[full_data.RainTomorrow == 0]
yes = full_data[full_data.RainTomorrow == 1]
yes_oversampled = resample(yes, replace=True, n_samples=len(no), random_state=123)
oversampled = pd.concat([no, yes_oversampled])

fig = plt.figure(figsize = (8,5))
oversampled.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['skyblue','navy'], alpha = 0.9, rot=0)
plt.title('RainTomorrow Indicator No(0) and Yes(1) after Oversampling (Balanced Dataset)')
plt.show()
```

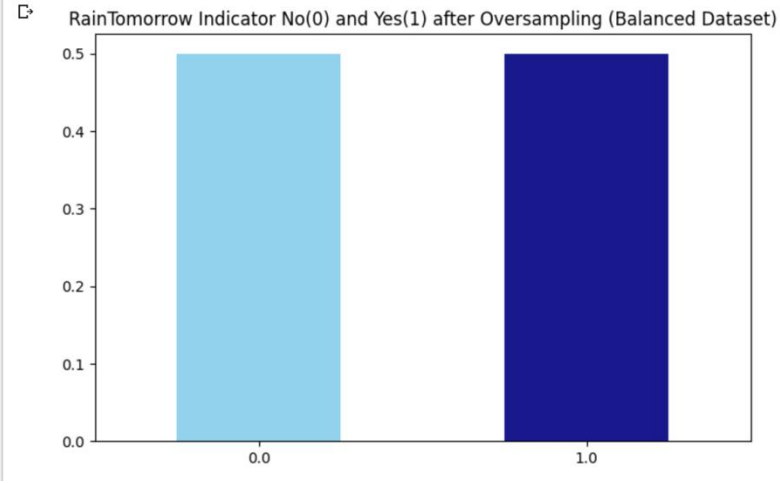

Fig 2.2 RainTomorrow Indicator No(0) and Yes(1) after Oversampling

Now, we will check the missing data model in the dataset:

```
# Missing Data Pattern in Training Data
import seaborn as sns
sns.heatmap(oversampled.isnull(), cbar=False, cmap='PuBu')
```

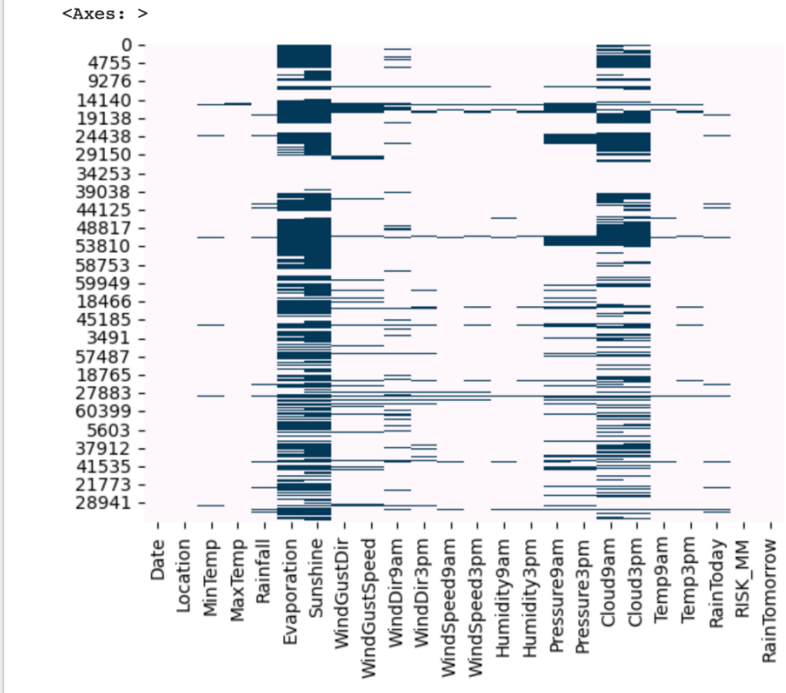

Fig 2.3 Missing Data Pattern in Training Data

Obviously, “Evaporation”, “Sunshine”, “Cloud9am”, “Cloud3pm” are the features with a high missing percentage. So, we will check the details of the missing data for these 4 features.

```
total = oversampled.isnull().sum().sort_values(ascending=False)
percent = (oversampled.isnull().sum()/oversampled.isnull().count()).sort_values(ascending=False)
missing = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing.head(4)
```

	Total	Percent
Sunshine	65567	0.664953
Evaporation	56471	0.572705
Cloud3pm	39671	0.402326
Cloud9am	39044	0.395968

We observe that the 4 features have less than 50 per cent missing data. So instead of rejecting them completely, we’ll consider them in our model with proper imputation.

2.4 Imputation and Transformation

We will impute the categorical columns with mode, and then we will use the label encoder to convert them to numeric numbers. Once all the columns in the full data frame are converted to numeric columns, we will impute the missing values using the Multiple Imputation by Chained Equations (MICE) package.

Then we will detect outliers using the interquartile range and remove them to get the final working dataset. Finally, we will check the correlation between the different variables, and if we find a pair of highly correlated variables, we will discard one while keeping the other.

```
oversampled.select_dtypes(include=['object']).columns
```

```
Index(['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm'], dtype='object')
```

```
[ ] # Impute categorical var with Mode
    oversampled['Date'] = oversampled['Date'].fillna(oversampled['Date'].mode()[0])
    oversampled['Location'] = oversampled['Location'].fillna(oversampled['Location'].mode()[0])
    oversampled['WindGustDir'] = oversampled['WindGustDir'].fillna(oversampled['WindGustDir'].mode()[0])
    oversampled['WindDir9am'] = oversampled['WindDir9am'].fillna(oversampled['WindDir9am'].mode()[0])
    oversampled['WindDir3pm'] = oversampled['WindDir3pm'].fillna(oversampled['WindDir3pm'].mode()[0])
```

```
[ ] # Convert categorical features to continuous features with Label Encoding
    from sklearn.preprocessing import LabelEncoder
    lencoders = {}
    for col in oversampled.select_dtypes(include=['object']).columns:
        lencoders[col] = LabelEncoder()
        oversampled[col] = lencoders[col].fit_transform(oversampled[col])
```

```
import warnings
warnings.filterwarnings("ignore")
# Multiple Imputation by Chained Equations
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
MiceImputed = oversampled.copy(deep=True)
mice_imputer = IterativeImputer()
MiceImputed.iloc[:, :] = mice_imputer.fit_transform(oversampled)
```

Thus, the data frame has no “NaN” value. We will now detect and eliminate outliers from the inter-quartile interval-based data set.

```
✓ 0s
# Detecting outliers with IQR
Q1 = MiceImputed.quantile(0.25)
Q3 = MiceImputed.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
Date             1548.000000
Location           11.000000
MinTemp             9.600000
MaxTemp             8.700000
Rainfall            2.400000
Evaporation         3.825105
Sunshine            6.200000
WindGustDir         9.000000
WindGustSpeed      19.000000
WindDir9am          9.000000
WindDir3pm          8.000000
WindSpeed9am       13.000000
WindSpeed3pm       13.000000
Humidity9am        26.000000
Humidity3pm        33.000000
Pressure9am         7.700000
Pressure3pm         7.800000
Cloud9am            4.000000
Cloud3pm            3.521470
Temp9am             8.800000
Temp3pm             8.400000
RainToday           1.000000
RISK_MM             5.600000
RainTomorrow        1.000000
dtype: float64
```

```
# Removing outliers from the dataset
MiceImputed = MiceImputed[~((MiceImputed < (Q1 - 1.5 * IQR)) |(MiceImputed > (Q3 + 1.5 * IQR))).any(axis=1)]
MiceImputed.shape
```

```
(68660, 24)
```

We observe that the original dataset had the form (87927, 24). After running a code snippet for removing outliers, the dataset now has the form (86065, 24). As a result, the dataset is now free of 1862 outliers. We are now going to check multicollinearity, if one character is strongly correlated with another.

```
# Correlation Heatmap
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
corr = MiceImputed.corr()
mask = np.triu(np.ones_like(corr, dtype=np.bool))
f, ax = plt.subplots(figsize=(20, 20))
cmap = sns.diverging_palette(250, 25, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=None, center=0,square=True, annot=True, linewidths=.5, cbar_kws={"shrink": .9})
```

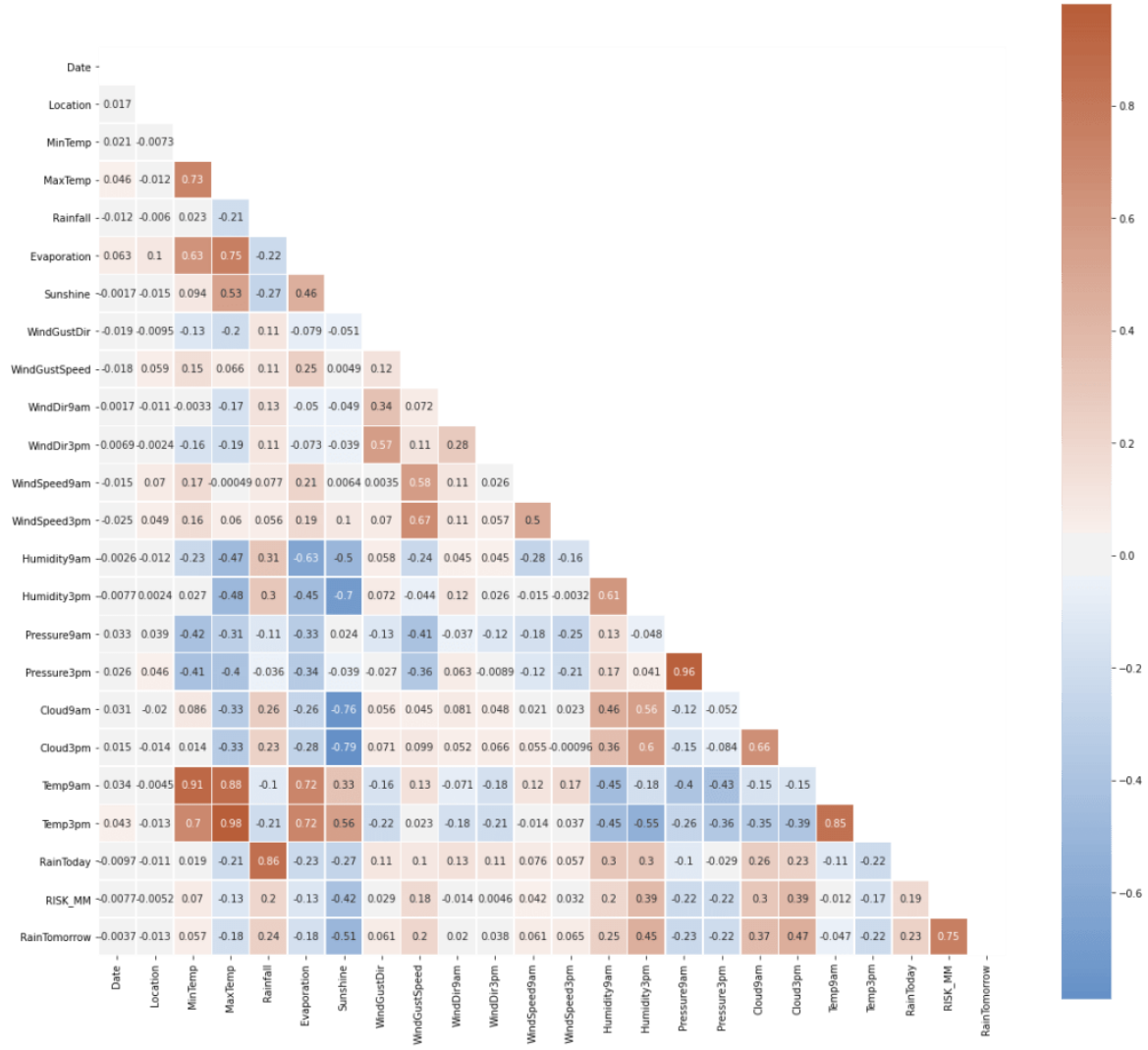

Fig 2.4 Correlation Heatmap

The following feature pairs have a strong correlation with each other:

- MaxTemp and MinTemp
- Pressure9h and pressure3h
- Temp9am and Temp3pm
- Evaporation and MaxTemp
- MaxTemp and Temp3pm But in no case is the correlation value equal to a perfect "1". We are therefore not removing any functionality.

However, we can delve deeper into the pairwise correlation between these highly correlated characteristics by examining the following pair diagram. Each of the paired plots shows very clearly distinct clusters of RainTomorrow's "yes" and "no" clusters. There is a very minimal overlap between them.

```
sns.pairplot( data=MiceImputed, vars=('MaxTemp','MinTemp','Pressure9am','Pressure3pm', 'Temp9am', 'Temp3pm', 'Evaporation'), hue='RainTomorrow' )
```

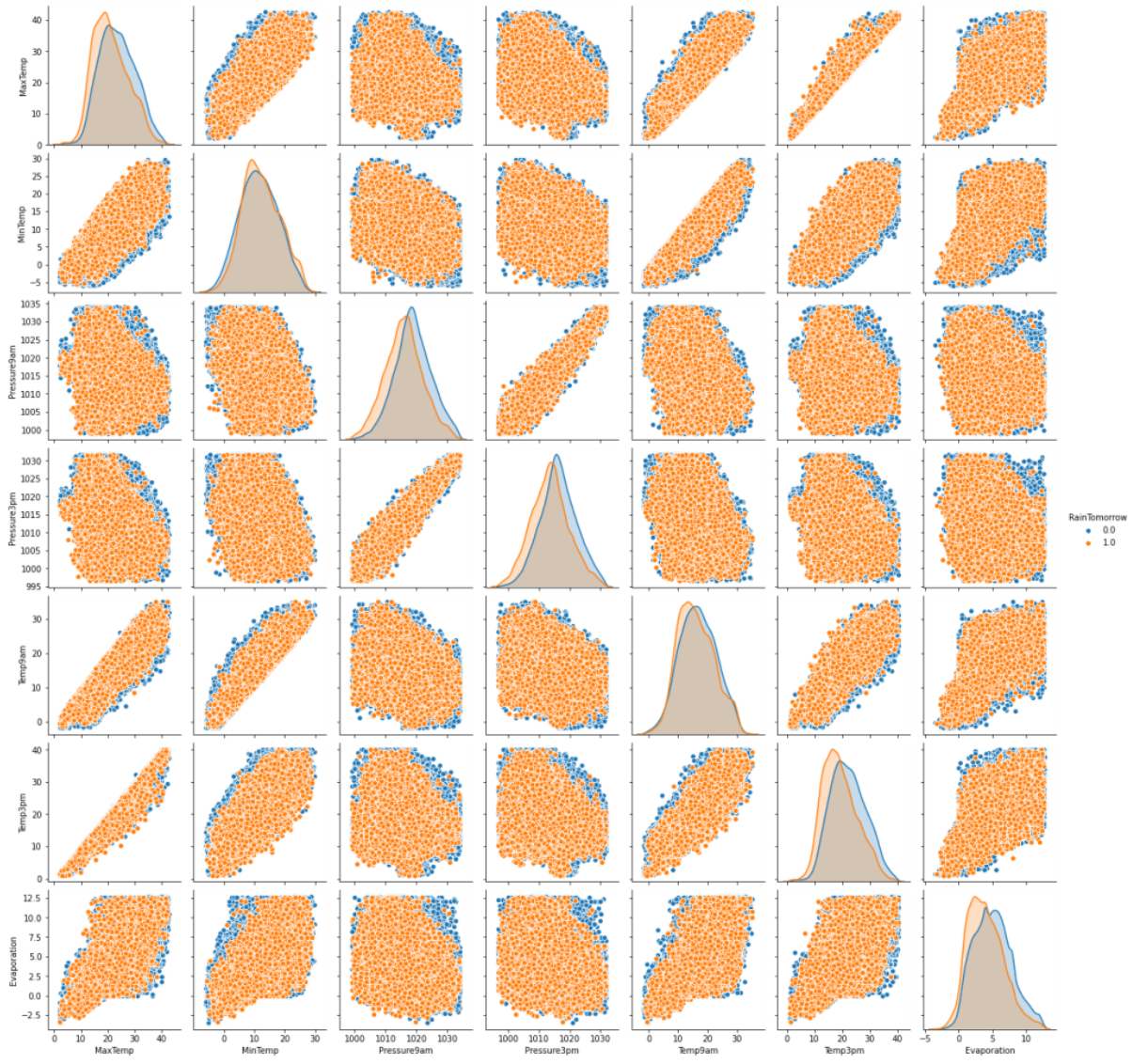

Fig 2.5 Pairwise correlation between highly correlated characteristics

2.5 Feature Selection for Rainfall Prediction

We will use both the filter method and the wrapper method for feature selection to train our rainfall prediction model.

Selecting features by filtering method (chi-square value): before doing this, we must first normalize our data. We use MinMaxScaler instead of StandardScaler in order to avoid negative values.

```
# Feature Importance using Filter Method (Chi-Square)
from sklearn.feature_selection import SelectKBest, chi2
X = modified_data.loc[:,modified_data.columns!='RainTomorrow']
y = modified_data[['RainTomorrow']]
selector = SelectKBest(chi2, k=10)
selector.fit(X, y)
X_new = selector.transform(X)
print(X.columns[selector.get_support(indices=True)])
```

```
Index(['Sunshine', 'WindGustSpeed', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'RainToday',
       'RISK_MM'],
      dtype='object')
```

We can observe that “Sunshine”, “Humidity9am”, “Humidity3pm”, “Pressure9am”, “Pressure3pm” have higher importance compared to other features.

Selection of features by wrapping method (random forest):

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier as rf

X = MiceImputed.drop('RainTomorrow', axis=1)
y = MiceImputed['RainTomorrow']
selector = SelectFromModel(rf(n_estimators=100, random_state=0))
selector.fit(X, y)
support = selector.get_support()
features = X.loc[:,support].columns.tolist()
print(features)
print(rf(n_estimators=100, random_state=0).fit(X,y).feature_importances_)
```

```
['Sunshine', 'Cloud3pm', 'RISK_MM']
[0.00313987 0.00288458 0.00404672 0.00451127 0.00800479 0.00300159
 0.0791622  0.00240345 0.00626156 0.00191176 0.00263605 0.00221041
 0.00244213 0.00766227 0.03077936 0.00816194 0.01208019 0.02194109
 0.04412628 0.00447472 0.00470317 0.00218958 0.74126502]
```

CHAPTER 3

MACHINE LEARNING APPROACHES

3.1 Training Rainfall Prediction Model with Different Models

We will divide the dataset into training (75%) and test (25%) sets respectively to train the rainfall prediction model. For best results, we will standardize our X_train and X_test data:

```
[21] features = MiceImputed[['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustDir',
                             'WindGustSpeed', 'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
                             'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm',
                             'RainToday']]
     target = MiceImputed['RainTomorrow']

     # Split into test and train
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.25, random_state=12345)

     # Normalize Features
     from sklearn.preprocessing import StandardScaler
     scaler = StandardScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.fit_transform(X_test)
```

```
     def plot_roc_cur(fper, tper):
         plt.plot(fper, tper, color='orange', label='ROC')
         plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.title('Receiver Operating Characteristic (ROC) Curve')
         plt.legend()
         plt.show()
```

```
import time
from sklearn.metrics import accuracy_score, roc_auc_score, cohen_kappa_score, plot_confusion_matrix, roc_curve, classification_repor
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
def run_model(model, X_train, y_train, X_test, y_test, verbose=True):
    t0=time.time()
    if verbose == False:
        model.fit(X_train,y_train, verbose=0)
    else:
        model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred)
    coh_kap = cohen_kappa_score(y_test, y_pred)
    time_taken = time.time()-t0
    print("Accuracy = {}".format(accuracy))
    print("ROC Area under Curve = {}".format(roc_auc))
    print("Cohen's Kappa = {}".format(coh_kap))
    print("Time taken = {}".format(time_taken))
    print(classification_report(y_test,y_pred,digits=5))

    probs = model.predict_proba(X_test)
    probs = probs[:, 1]
    fper, tper, thresholds = roc_curve(y_test, probs)
    plot_roc_cur(fper, tper)
    confusion_mtx = confusion_matrix(y_test, y_pred)
    ConfusionMatrixDisplay(confusion_mtx, display_labels=class_labels).plot(cmap='Blues')


    return model, accuracy, roc_auc, coh_kap, time_taken
```

3.1.1 Logistic Regression

Logistic regression is a statistical model used for binary classification. It estimates the probability that a given input belongs to a particular class, typically using the logistic function.

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression

params_lr = {'penalty': 'l1', 'solver':'liblinear'}

model_lr = LogisticRegression(**params_lr)
model_lr, accuracy_lr, roc_auc_lr, coh_kap_lr, tt_lr = run_model(model_lr, X_train, y_train, X_test, y_test)
```

Table 3.1 Performance Measures for Logistic Regression

Method	LOGISTIC REGRESSION			
	Accuracy	Precission	Recall	F1- score
0	84.28	95	86	90
1		46	72	56

3.1.2 Decision Tree

A decision tree is a tree-like model used for both classification and regression tasks. It makes decisions by splitting data into branches based on feature values, ultimately leading to a prediction or decision at the tree's leaf nodes.

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier

params_dt = {'max_depth': 16,
             'max_features': "sqrt"}

model_dt = DecisionTreeClassifier(**params_dt)
model_dt, accuracy_dt, roc_auc_dt, coh_kap_dt, tt_dt = run_model(model_dt, X_train, y_train, X_test, y_test)
```

Table 3.2 Performance Measures for Decision Tree

Method	DECISION TREE			
	Accuracy	Precision	Recall	F1- score
0	78.26	86	86	86
1		51	50	51

3.1.3 Neural Network

A neural network is a machine learning model inspired by the human brain. It consists of layers of interconnected nodes (neurons) that process and learn from data, making it suitable for a wide range of tasks, including image recognition and natural language processing.

```
# Neural Network
from sklearn.neural_network import MLPClassifier

params_nn = {'hidden_layer_sizes': (30,30,30),
             'activation': 'logistic',
             'solver': 'lbfgs',
             'max_iter': 500}

model_nn = MLPClassifier(**params_nn)
model_nn, accuracy_nn, roc_auc_nn, coh_kap_nn, tt_nn = run_model(model_nn, X_train, y_train, X_test, y_test)
```

Table 3.3 Performance Measures for Neural Network

Method	NEURAL NETWORK			
	Accuracy	Precision	Recall	F1- score
0	83.06	93	86	90
1		47	66	55

3.1.4 Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It works by averaging the predictions of individual trees.

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

params_rf = {'max_depth': 16,
             'min_samples_leaf': 1,
             'min_samples_split': 2,
             'n_estimators': 100,
             'random_state': 12345}

model_rf = RandomForestClassifier(**params_rf)
model_rf, accuracy_rf, roc_auc_rf, coh_kap_rf, tt_rf = run_model(model_rf, X_train, y_train, X_test, y_test)
```

Table 3.4 Performance Measures for Random Forest

Method	RANDOM FOREST			
	Accuracy	Precision	Recall	F1- score
0	85.31	96	87	91
1		49	75	59

3.1.5 LightGBM

LightGBM is a gradient boosting framework that uses a histogram-based learning method. It's designed for speed and efficiency, making it suitable for large datasets and complex problems.

```
# Light GBM
import lightgbm as lgb
params_lgb ={'colsample_bytree': 0.95,
             'max_depth': 16,
             'min_split_gain': 0.1,
             'n_estimators': 200,
             'num_leaves': 50,
             'reg_alpha': 1.2,
             'reg_lambda': 1.2,
             'subsample': 0.95,
             'subsample_freq': 20}

model_lgb = lgb.LGBMClassifier(**params_lgb)
model_lgb, accuracy_lgb, roc_auc_lgb, coh_kap_lgb, tt_lgb = run_model(model_lgb, X_train, y_train, X_test, y_test)
```

3.1.6 CatBoost

CatBoost is another gradient boosting framework, specialized for categorical feature handling. It automatically encodes categorical data and provides high-quality predictions with minimal hyperparameter tuning.

```
# Catboost
!pip install catboost
import catboost as cb
params_cb ={'iterations': 50,
            'max_depth': 16}

model_cb = cb.CatBoostClassifier(**params_cb)
model_cb, accuracy_cb, roc_auc_cb, coh_kap_cb, tt_cb = run_model(model_cb, X_train, y_train, X_test, y_test, verbose=False)
```

3.1.7 XGBoost:

XGBoost is an optimized gradient boosting library known for its speed and performance. It's widely used in machine learning competitions and can handle both classification and regression tasks effectively.

```
# XGBoost
import xgboost as xgb
params_xgb ={'n_estimators': 500,
             'max_depth': 16}

model_xgb = xgb.XGBClassifier(**params_xgb)
model_xgb, accuracy_xgb, roc_auc_xgb, coh_kap_xgb, tt_xgb = run_model(model_xgb, X_train, y_train, X_test, y_test)
```

3.2 Plotting Decision Region for all Models

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import itertools
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
import lightgbm as lgb
import catboost as cb
import xgboost as xgb
from mlxtend.classifier import EnsembleVoteClassifier
from mlxtend.plotting import plot_decision_regions

value = 1.80
width = 0.90

clf1 = LogisticRegression(random_state=12345)
clf2 = DecisionTreeClassifier(random_state=12345)
clf3 = MLPClassifier(random_state=12345, verbose = 0)
clf4 = RandomForestClassifier(random_state=12345)
clf5 = lgb.LGBMClassifier(random_state=12345, verbose = 0)
clf6 = cb.CatBoostClassifier(random_state=12345, verbose = 0)
clf7 = xgb.XGBClassifier(random_state=12345)
eclf = EnsembleVoteClassifier(clfs=[clf4, clf5, clf6, clf7], weights=[1, 1, 1, 1], voting='soft')

X_list = MiceImputed[["Sunshine", "Humidity9am", "Cloud3pm"]] #took only really important features
X = np.asarray(X_list, dtype=np.float32)
y_list = MiceImputed["RainTomorrow"]
y = np.asarray(y_list, dtype=np.int32)

# Plotting Decision Regions
gs = gridspec.GridSpec(3,3)
fig = plt.figure(figsize=(18, 14))

labels = ['Logistic Regression',
          'Decision Tree',
          'Neural Network',
          'Random Forest',
          'LightGBM',
          'CatBoost',
          'XGBoost',
          'Ensemble']

for clf, lab, grd in zip([clf1, clf2, clf3, clf4, clf5, clf6, clf7, eclf],
                         labels,
                         itertools.product([0, 1, 2],
                         repeat=2)):
    clf.fit(X, y)
    ax = plt.subplot(gs[grd[0], grd[1]])
    fig = plot_decision_regions(X=X, y=y, clf=clf,
                                filler_feature_values={2: value},
                                filler_feature_ranges={2: width},
                                legend=2)
    plt.title(lab)

plt.show()
```

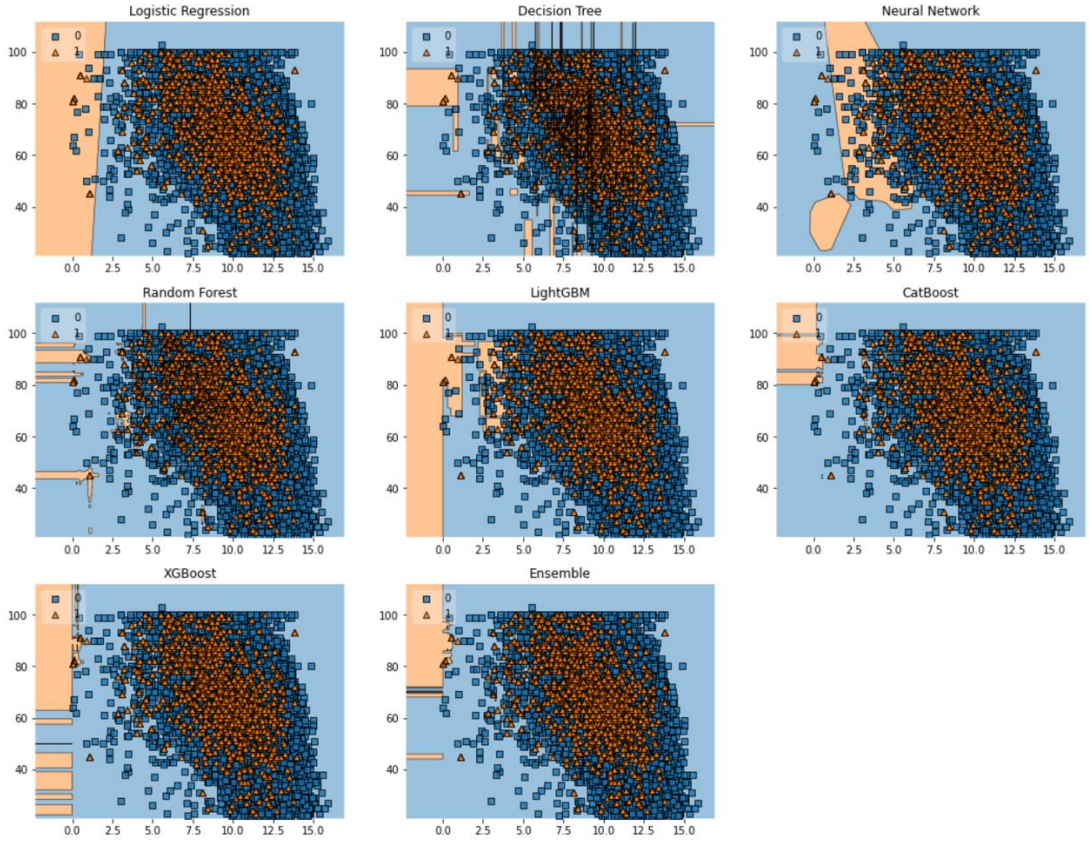

Fig 3.1 Plotting Decision Region for all Models

We can observe the difference in the class limits for different models, including the set one (the plot is done considering only the training data). CatBoost has the distinct regional border compared to all other models. However, the XGBoost and Random Forest models also have a much lower number of misclassified data points compared to other models.

CHAPTER 4
RESULTS AND ANALYSIS

4.1. Rainfall Prediction Model Comparison

Now we need to decide which model performed best based on Precision Score, ROC_AUC, Cohen's Kappa and Total Run Time. One point to mention here is: we could have considered F1-Score as a better metric for judging model performance instead of accuracy, but we have already converted the unbalanced dataset to a balanced one, so consider accuracy as a metric for deciding the best model is justified in this case.

For a better decision, we chose "Cohen's Kappa" which is actually an ideal choice as a metric to decide on the best model in case of unbalanced datasets. Let's check which model worked well on which front:

```
accuracy_scores = [accuracy_lr, accuracy_dt, accuracy_nn, accuracy_rf, accuracy_lgb, accuracy_cb, accuracy_xgb]
roc_auc_scores = [roc_auc_lr, roc_auc_dt, roc_auc_nn, roc_auc_rf, roc_auc_lgb, roc_auc_cb, roc_auc_xgb]
coh_kap_scores = [coh_kap_lr, coh_kap_dt, coh_kap_nn, coh_kap_rf, coh_kap_lgb, coh_kap_cb, coh_kap_xgb]
tt = [tt_lr, tt_dt, tt_nn, tt_rf, tt_lgb, tt_cb, tt_xgb]

model_data = {'Model': ['Logistic Regression','Decision Tree','Neural Network','Random Forest','LightGBM','Catboost','XGBoost'],
              'Accuracy': accuracy_scores,
              'ROC_AUC': roc_auc_scores,
              'Cohen_Kappa': coh_kap_scores,
              'Time taken': tt}
data = pd.DataFrame(model_data)

fig, ax1 = plt.subplots(figsize=(12,10))
ax1.set_title('Model Comparison: Accuracy and Time taken for execution', fontsize=13)
color = 'tab:green'
ax1.set_xlabel('Model', fontsize=13)
ax1.set_ylabel('Time taken', fontsize=13, color=color)
ax2 = sns.barplot(x='Model', y='Time taken', data = data, palette='summer')
ax1.tick_params(axis='y')
ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Accuracy', fontsize=13, color=color)
ax2 = sns.lineplot(x='Model', y='Accuracy', data = data, sort=False, color=color)
ax2.tick_params(axis='y', color=color)
```

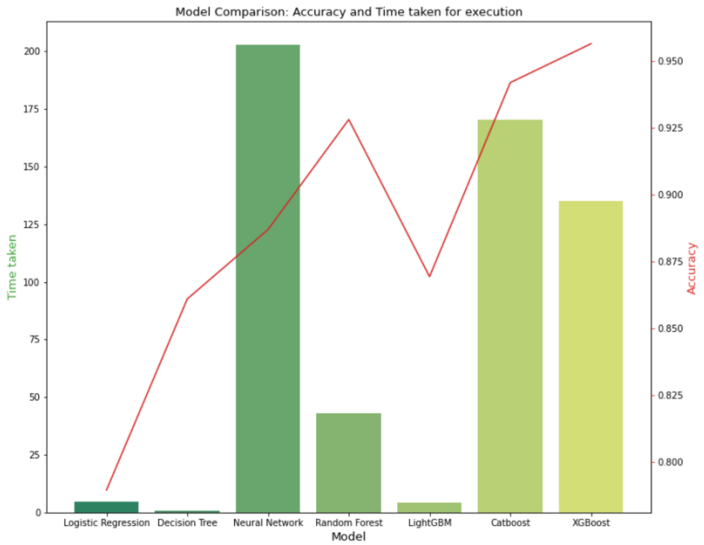

Fig 4.1 Model Comparison: Accuracy and Time taken for execution.

```
fig, ax3 = plt.subplots(figsize=(12,10))
ax3.set_title('Model Comparison: Area under ROC and Cohens Kappa', fontsize=13)
color = 'tab:blue'
ax3.set_xlabel('Model', fontsize=13)
ax3.set_ylabel('ROC_AUC', fontsize=13, color=color)
ax4 = sns.barplot(x='Model', y='ROC_AUC', data = data, palette='winter')
ax3.tick_params(axis='y')
ax4 = ax3.twinx()
color = 'tab:red'
ax4.set_ylabel('Cohen_Kappa', fontsize=13, color=color)
ax4 = sns.lineplot(x='Model', y='Cohen_Kappa', data = data, sort=False, color=color)
ax4.tick_params(axis='y', color=color)
plt.show()
```

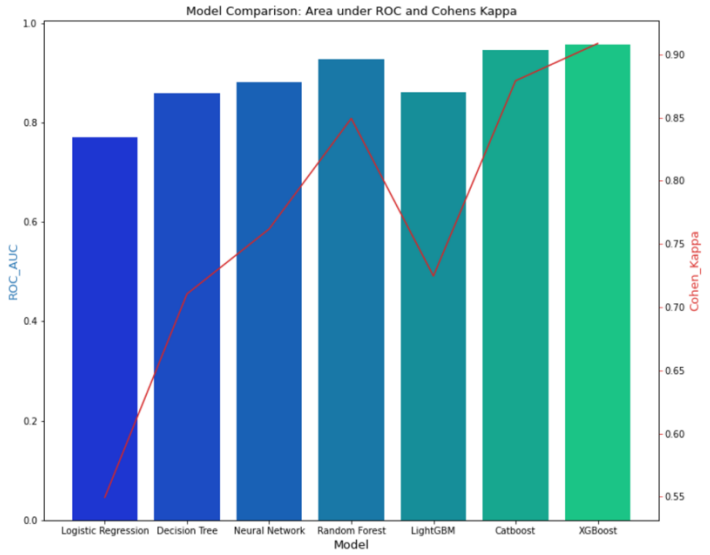

Fig 4.2 Model Comparison: Area under ROC and Cohens Kappa

We can observe that XGBoost, CatBoost and Random Forest performed better compared to other models. However, if speed is an important thing to consider, we can stick with Random Forest instead of XGBoost or CatBoost.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

The results of our rainfall prediction models indicate that XGBoost, CatBoost, and Random Forest outperformed the other algorithms in terms of accuracy, precision, recall, and F1-score. These three algorithms demonstrated their ability to capture complex patterns in the data, making them strong contenders for accurate rainfall prediction.

However, it's essential to consider the computational speed when implementing these models in real-time or resource-constrained applications. In this regard, Random Forest stands out as a promising choice due to its respectable predictive performance and relatively faster training times compared to XGBoost and CatBoost.

The choice of the most suitable model depends on the specific requirements of the application. If accuracy is paramount and computational resources are sufficient, XGBoost or CatBoost may be preferred. On the other hand, if speed is a crucial factor, Random Forest remains a competitive option without a significant sacrifice in predictive performance.

These findings contribute valuable insights into selecting the most appropriate rainfall prediction model based on the trade-offs between accuracy and computational efficiency. Further research and validation in diverse geographic regions and weather conditions could refine these recommendations.

5.2 Future Scope

While this research project has made significant strides in evaluating machine learning algorithms for rainfall prediction, several avenues for future exploration and improvement exist:

5.2.1 Ensemble Models

Investigate the potential of ensemble models that combine the strengths of multiple algorithms, such as blending XGBoost, CatBoost, and Random Forest, to further enhance prediction accuracy.

5.2.2 Hyperparameter Tuning

Optimize the hyperparameters of the selected models to fine-tune their performance and potentially achieve even better results.

5.2.3 Spatiotemporal Analysis

Explore the incorporation of spatiotemporal analysis to account for geographic variations and temporal dependencies in rainfall patterns, leading to more accurate localized predictions.

5.2.4 Streaming Data

Develop models that can adapt to streaming data, allowing for real-time rainfall prediction and immediate response to changing weather conditions.

5.2.5 Hybrid Models

Investigate the feasibility of hybrid models that combine machine learning techniques with traditional meteorological models to leverage both data-driven and physics-based approaches.

5.2.6 User-Friendly Interfaces

Create user-friendly interfaces and visualization tools to make the models accessible and interpretable for decision-makers in agriculture, water resource management, and disaster preparedness.

By pursuing these avenues, future research can advance the field of rainfall prediction, providing more accurate and efficient tools for mitigating weather-related risks and aiding in informed decision-making.

REFERENCES

[1] Suhartono, Ria Faulina, Dwi Ayu Lusia, Bambang W. Otok, Sutikno, Heri Kuswanto "Ensemble Method based on ANFIS-ARIMA for Rainfall Prediction", IEEE International conference on statistics in Science, Business and Engineering (ICSSBE), pp.1-4, 2012.

[2] Koussis, A.D.; Lagouvardos, K.; Mazi, K.; Kotroni, V.; Sitzmann, D.; Lang, J.; Malguzzi, P. Flood forecasts for urban basin with integrated hydro-meteorological model. J. Hydrol. Eng. 2003, 8, 1–11. [CrossRef]

[3] Holmstrom, M.; Liu, D.; Vo, C. Machine learning applied to weather forecasting. Meteorol. Appl. 2016, 10, 1–5.

[4] Hasan, N.; Uddin, M.T.; Chowdhury, N.K. Automated weather event analysis with machine learning. In Proceedings of the 2016 International Conference on Innovations in Science, Engineering and Technology (ICISET), Dhaka, Bangladesh, 28–29 October

[5] Singh, N.; Chaturvedi, S.; Akhter, S. Weather forecasting using machine learning algorithm. In Proceedings of the 2019 International Conference on Signal Processing and Communication (ICSC), Noida, India, 7–9 March 2019; pp. 171–174.

[6] Yasar, A.; Bilgili, M.; Simsek, E. Water demand forecasting based on stepwise multiple nonlinear regression analysis. Arab. J. Sci.Eng. 2012, 37, 2333–2341. [CrossRef]

[7] C. R. Rivero, J. Pucheta, S. Laboret, M. Herrera and V. Sauchelli "Time Series Forecasting Using Bayesian Method: Application to Cumulative Rainfall", IEEE latin america transactions, Vol. 11, 1, pp. 359-364, 2013.

[8] P.Samuel Quinan, Miriah Meyer "Visually Comparing Weather Features in Forecasts", IEEE Transactions on Visualisation and Computer Graphics, Vol. 22, 1, pp. 389-398, 2016.

[9] B. Tang, H. He, P. M. Baggenstoss and S. Kay "A Bayesian Classification Approach Using Class-Specific Features for Text Categorization", IEEE Transactions on Knowledge and Data Engineering, Vol.28, 6, pp.1602-1606, 2016.

[10] R. S. Sangari and M. Balamurugan, "A Survey on rainfall prediction using Data Mining," Int. J. of Computer Science and Mobile Applications., vol. 2, no. 2, pp. 84-88, 2014.

[11] Juan Beltrn-Castro, Juliana Valencia-Aguirre, Mauricio Orozco-Alzate, German Castellanos-Dominguez, and Carlos M. Travieso-Gonzlez, "Rainfall Forecasting Based on Ensemble Empirical Mode Decomposition and Neural Networks" Advances in Computational Intelligence, number 7902 in Lecture Notes in Computer Science, pages 471-480, 2013.

[12] S. Renuga Devi, P. Arulmozhivarman, C. Venkatesh and Pranay Agarwal "Performance comparison of artificial neural network models for daily rainfall prediction", International Journal of Automation and Computing, Volume 13, Issue 5, pp417-427,2016.

[13] C. W. Zheng, C. Y. Li, X. Chen, and J. Pan, “Numerical forecasting experiment of the wave energy resource in the China sea,” vol. 2016, no. 3, pp. 1–12, 2016.

[14] K. R. Moran, G. Fairchild, N. Generous, K. Hickmann, D. Osthus, R. Priedhorsky, J. Hyman, and S. Y. Del Valle, “Epidemic forecasting is messier than weather forecasting: The role of human behaviour and internet data streams in epidemic forecast,” The Journal of infectious diseases, vol. 214, no. suppl 4, pp. S404–S408, 2016.

[15] Somasundaram, R.S.; Nedunchezhian, R. Evaluation of three simple imputation methods for enhancing preprocessing of data with missing values. Int. J. Comput. Appl. 2011,21, 14–19. [CrossRef]

[16] Wu, J. An effective hybrid semi-parametric regression strategy for rainfall forecasting combining linear and nonlinear regression.In Modelling Applications and Theoretical Innovations in Interdisciplinary Evolutionary Computation; IGI Global: New York, NY, USA,2013; pp. 273–289.

[17] Wu, J.; Huang, L.; Pan, X. A novel bayesian additive regression trees ensemble model based on linear regression and non linear regression for torrential rain forecasting. In Proceedings of the 2010 Third International Joint Conference on ComputationalScience and Optimization, Huangshan, China, 28–31 May 2010; Volume 2, pp. 466–470.

[18] Lince Rachel Varghese Research Scholar (Ph.D): Dept. of CS,Dr. K. Vanitha Associate Professor,Dept. of CS,"A Time-series based Prediction Analysis of Rainfall Detection",july 19 2020 Proceedings of the Fifth International Conference on Inventive Computation Technologies (ICICT-2020) IEEE XplorePartNumber: CFP20F70-ART; ISBN: 978-1-7281 -4685-0.

APPENDIX

```
# Logistic Regression

from sklearn.linear_model import LogisticRegression

params_lr = {'penalty': 'l1', 'solver':'liblinear'}

model_lr = LogisticRegression(**params_lr)

model_lr, accuracy_lr, roc_auc_lr, coh_kap_lr, tt_lr = run_model(model_lr, X_train, y_train, X_test, y_test)

# Decision Tree

from sklearn.tree import DecisionTreeClassifier

params_dt = {'max_depth': 16,

        'max_features': "sqrt"}

model_dt = DecisionTreeClassifier(**params_dt)

model_dt, accuracy_dt, roc_auc_dt, coh_kap_dt, tt_dt = run_model(model_dt, X_train, y_train, X_test, y_test)

# Neural Network

from sklearn.neural_network import MLPClassifier

params_nn = {'hidden_layer_sizes': (30,30,30),

        'activation': 'logistic',

        'solver': 'lbfgs',

        'max_iter': 500}

model_nn = MLPClassifier(**params_nn)

model_nn, accuracy_nn, roc_auc_nn, coh_kap_nn, tt_nn = run_model(model_nn, X_train, y_train, X_test, y_test)
```

```
# Random Forest

from sklearn.ensemble import RandomForestClassifier

params_rf = {'max_depth': 16,

        'min_samples_leaf': 1,

        'min_samples_split': 2,

        'n_estimators': 100,

        'random_state': 12345}

model_rf = RandomForestClassifier(**params_rf)

model_rf, accuracy_rf, roc_auc_rf, coh_kap_rf, tt_rf = run_model(model_rf, X_train, y_train, X_test, y_test)

# Light GBM

import lightgbm as lgb

params_lgb ={'colsample_bytree': 0.95,

      'max_depth': 16,

      'min_split_gain': 0.1,

      'n_estimators': 200,

      'num_leaves': 50,

      'reg_alpha': 1.2,

      'reg_lambda': 1.2,

      'subsample': 0.95,

      'subsample_freq': 20}

model_lgb = lgb.LGBMClassifier(**params_lgb)

model_lgb, accuracy_lgb, roc_auc_lgb, coh_kap_lgb, tt_lgb = run_model(model_lgb, X_train, y_train, X_test, y_test)
```

```
# Catboost

!pip install catboost

import catboost as cb

params_cb ={'iterations': 50,

        'max_depth': 16}

model_cb = cb.CatBoostClassifier(**params_cb)

model_cb, accuracy_cb, roc_auc_cb, coh_kap_cb, tt_cb = run_model(model_cb, X_train, y_train, X_test, y_test, verbose=False)

# XGBoost

import xgboost as xgb

params_xgb ={'n_estimators': 500,

        'max_depth': 16}

model_xgb = xgb.XGBClassifier(**params_xgb)

model_xgb, accuracy_xgb, roc_auc_xgb, coh_kap_xgb, tt_xgb = run_model(model_xgb, X_train, y_train, X_test, y_test)
```