Rainfall Prediction using Feature Engineering Techniques

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

P. Satish Sarma	**Inturi Harshith**	**Vanganur Madhusudhan**
18BEC0397	**18BEC0389**	**18BEC0951**

Under the guidance of

Dr. Karthikeyan M
School of Electronics Engineering
VIT, Vellore

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

April 2022

DECLARATION

We hereby declare that the thesis entitled “Rainfall Prediction using Feature Engineering Techniques" submitted by us, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering* to VIT is a record of bonafide work carried out by us under the supervision of Dr. Karthikeyan M.

We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore
Date : 30th April 2022

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled “Rainfall Prediction using Feature Engineering Techniques” submitted by P. Satish Sarma (18BEC0397), Inturi Harshith (18BEC0389), and Vanganur Madhusudhan (18BEC0951), School of Electronics Engineering, VIT, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering*, is a record of bonafide work carried out by them under my supervision during the period, 01.12.2022 to 30.04.2022, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore
Date : 30th April 2022

Signature of the Candidate

Internal Examiner **External Examiner**

Head of the Department
Electronics and Communication

ACKNOWLEDGEMENT

We take this opportunity to express our heartfelt gratitude to our project guide Dr. Karthikeyan M whose exceptional guidance helped us in the development of our proposed system for our problem statement. We shall remain forever indebted for his constructive criticism, clear thinking, keen interest, constant encouragement and forbearing right from the beginning of this research to its completion. We are extremely grateful for various resources provided to us for making our project easier and for making sure that the quality of the project was not compromised. Finally, we would like to thank VIT University for incorporating this final year capstone project as a part of our curriculum, for it has provided us with a wonderful experience of learning and discovery.

Satish Sarma Palepu

Inturi Harshith

Vanganur Madhusudhan

EXECUTIVE SUMMARY

A reliable prediction of rainfall levels on a day-to-day basis is extremely beneficial to the water and agricultural supply of a country, as it provides ease of management and effective pre-planning of water usage. While the existing systems do provide good predictions, there is clear scope for improvement in various aspects of these systems. There exist some areas such as time and power consumption, and complexity of the model, which can be better balanced. Our project attempts to improve upon such aspects, using specific techniques of data manipulation and analysis, to create a more reliable prediction. This design thus provides an effective rainfall prediction model of the sort, which strikes a good balance of high reliability, low computational power necessity, and reasonable time consumption. With a simple yet detailed analysis of the various factors contributing to rainfall, our design is able to provide predictions with optimal error, making it an excellent tool for predicting rainfall in an area.

Index

List of Figures

List of Tables

List of Abbreviations

IoT	-	Internet of Things
ML	-	Machine Learning
EL	-	Ensemble Learning
LR	-	Linear Regression
RF	-	Random Forest
DT	-	Decision Trees
BMC	-	Bayesian Model Combination
LSTM	-	Long Short-Term Memory
ConvNet	-	Convolutional Neural Network
XGBoost	-	Extreme Gradient Boost
GBM	-	Gradient Boost Machine
SVM	-	Support Vector Machines
GPU	-	Graphic Processing Unit
CPU	-	Central Processing Unit
VIF	-	Variable Inflation Factor
MAE	-	Mean Absolute Error
RMSE	-	Root Mean Square Error

Symbols and Notations

Δ	-	Absolute Difference
Σ	-	Summation
$\surd$	-	Square Root
$\|m\|$	-	Absolute Value of ‘m’
ε	-	Random Error

1. Introduction

a. Objective

Rainfall Prediction Systems are one of the fastest and most reliable procedures to predict rainfall distributions and screen floods. The systems use Internet of Things (IoT) devices to obtain data, and combine this data with Machine Learning and Data Engineering Techniques for the estimation of the amount of rainfall. Our primary objective in this work is to design and demonstrate a highly effective rainfall prediction system, and compare it to the existing works to show an improvement in its reliability and efficiency.

b. Motivation

There exist many methods of early detection of floods that use various machine learning models and prediction techniques to best estimate rainfall levels. However, they tend to be more time consuming, less efficient, and less effective than what is possible. The Data refining and Feature Engineering techniques carried out in our project are extremely effective in preparing the data to be clean and uniformly distributed, thus making our work less complex and more reliable

c. Background

The past few years have shown significant improvements in the development of highly efficient rainfall prediction systems. There are systems using various ensemble learning Bayesian models, SVMs, Deep Learning and convolutional neural network architectures to obtain highly reliable predictions. The systems range from being simple linear predictions to extremely complex and demanding algorithms which offer high accuracy in predictions. We aim to design a process which can offer a prediction with a very reasonable execution time, which need not necessarily require high computational power, and which is also very reliable in providing optimal error values

2. Project Description

The target of our work is to use various Data Preprocessing and Feature Engineering techniques to transform and shape the data collected from atmospheric factors such as temperature, atmospheric pressure, wind speed, etc., and feed the processed data into three Machine Learning algorithms - Linear Regression, Random Forest, and XGBoost, each of which predict the occurrence of rainfall in a specific area. Then, we derive a comparison of the performance of each algorithm, and arrive at a conclusion of which algorithm is best suited and most reliable for the prediction. Finally, we perform an analysis of various clusters in the area using k-means clustering algorithm, to determine the best suited prediction algorithm for each cluster, helping us to determine the best prediction technique overall with higher certainty.

3. Technical Specifications

The various softwares and hardware components used in our work are listed as follows:

- *Python* as the primary programming language used.
- *Jupyter Notebooks* as the primary programming software used.
- *Tableau* as the data visualization software for viewing maps of rainfall distributions and cluster analysis.
- *A Graphic Processing Unit (GPU)* in a computer, to compare the execution times of the model with standard CPU execution times.

We use the dataset of Rainfall in Australia from kaggle to obtain our raw data that we use in preprocessing.

4. Literature Survey

Author	Year	Title	Objective	Method Used	Conclusion
V P, Tharun & Prakash, Ramya & Subramanian, Renuga devi	2018	Prediction of Rainfall Using Data Mining Techniques	To predict rainfall via various Data mining techniques and machine learning models	Data Preprocessing, feeding into SVM, RF and DT regression models, and measuring performance of each	Best regression technique is RF with r-square value 0.981 and adjusted r-square value 0.980
Aswin S, Geetha P, Vinayakumar R	2018	Deep Learning Models for the Prediction of Rainfall	To obtain rainfall precipitation models using Deep Learning Architectures and their comparison	LSTM and ConvNet Deep Learning Architectures, and a comparison of their RMSE and MAE Error values	Both architectures effective, ConvNet architecture showing lower RMSE values
Dai, Weijun & Tang, Yanni & Zhang, Zeyu & Cai, Zhiming	2021	Ensemble Learning Technology for Coastal Flood Forecasting in Internet of Things Enabled Smart City	To use an ensemble learning method based on BMC-EL to predict floods	BMC-EL, LR, LSVM, QSVM, BPNN and RF models, and their reliability comparison	BMC-EL model has the best prediction results, and the highest reliability
M. Shankar, T. J. John, S. Karthick, B. Pattanaik, M. Pattnaik and S. Karthikeyan	2021	Internet of Things based Smart Flood forecasting and Early Warning System	To demonstrate an effective flood detection and early warning system using Arduino and IoT Systems	An IoT multi-sensor system to measure real-time environmental variables via cloud and monitor data threshold levels	IoT system gives productive and precise detection information for observing and cautioning purposes

5. Design Approach and Details

a. Flow Diagram

The working flow diagram of our project can be shown clearly as follows:

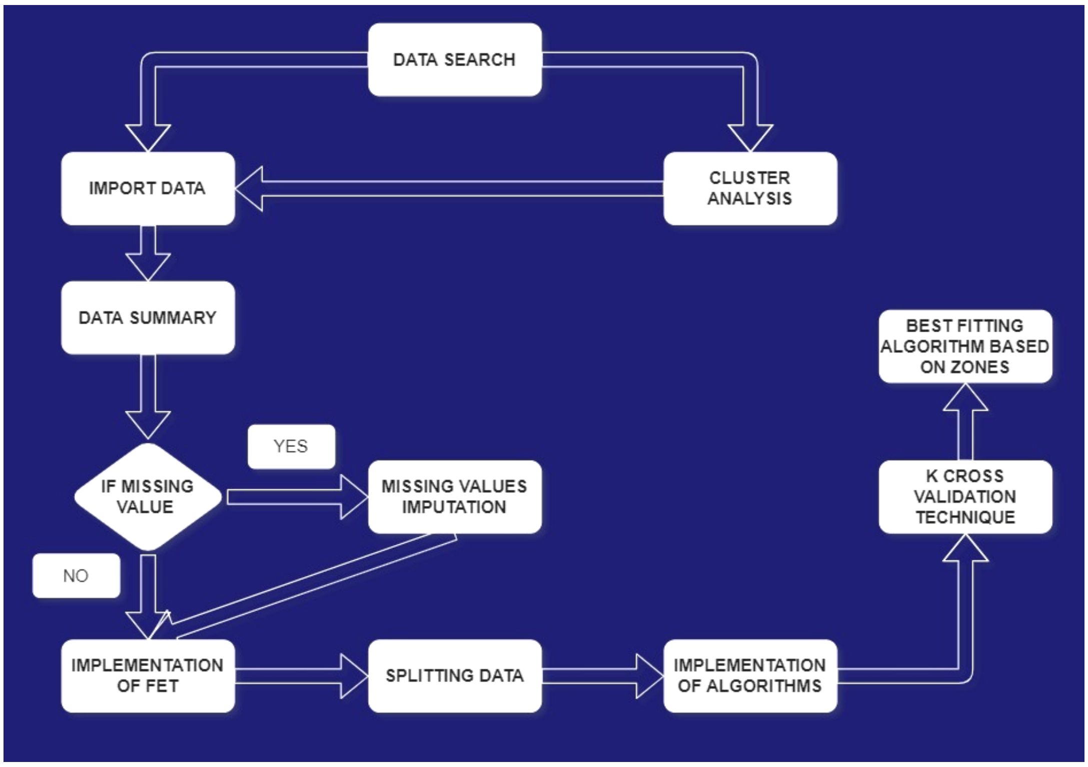

DATA SEARCH
IMPORT DATA
CLUSTER ANALYSIS
DATA SUMMARY
BEST FITTING ALGORITHM BASED ON ZONES
YES
IF MISSING VALUE
MISSING VALUES IMPUTATION
K CROSS VALIDATION TECHNIQUE
NO
IMPLEMENTATION OF FET
SPLITTING DATA
IMPLEMENTATION OF ALGORITHMS

Figure 1 - Dynamic Flow Diagram of the designed rainfall prediction model

b. Working Methodology

➔ The raw data is collected from the dataset "Rain in Australia" from kaggle. The dataset is imported into jupyter notebooks along with a collection of various python libraries used for data analysis and machine learning. Then, the following Feature Engineering techniques are used to shape the data accordingly.

➔ <u>Feature Engineering Techniques</u>:

The feature engineering techniques implemented are described as follows:

- *Imputation* - Imputation is used when there are any missing values or data present in the dataset, and they can be filled in without needing to remove and reduce the size of the dataset.

  Mean/median missing value imputation replaces the missing spaces with the calculated mean/median of the non-missing values in that category.

Most frequent categorical value imputation works with categorical features, a.k.a. The other values in the category, by replacing missing data with the most frequent values within each row or column.

- *Data Normalization* - Normalization is the process of scaling the data in the dataset to fit the values into a smaller range. It is useful when different attributes in the data are in different scales, which affects the overall efficiency of the algorithm, especially when the scales are significantly high in difference. Normalization fits all the sata to one simple scale, such as (0 to 1), or (-1 to 1).

- *Label Encoding* - The various labels of different variables in the data are converted from words to unique numbers, to convert them into a more machine-readable form. The categories are usually numbered 0,1,2,etc.

- *Resolving Multicollinearity Issues using VIF* - When multiple independent variables are highly correlated in the data, the effects of each variable are hard to analyze. This is called multicollinearity, and we resolve this issue using *Variable Inflation Factor (VIF)* which is a number that shows the strength of correlation between the highly correlated variables. If two values are found to be highly correlated through the comparison of their VIFs, one of the two values can be removed from the data to resolve the multicollinearity.

➔ <u>Machine Learning Algorithms</u>:

Three machine learning algorithms are implemented on the processed data, each of which is described as follows:

- *Linear Regression Algorithm* - Based on supervised learning techniques, this algorithm performs a regression task to map an independent variable in the data in a linear way to our variable which is to be predicted, to obtain our desired prediction.

- *Random Forest Algorithm* - This is a supervised learning algorithm which provides best results with regression and classification problems. It involves the combination of multiple decision trees on different parts of the dataset, and considering the average of the outputs of each to obtain our final output.

- *XGBoost Algorithm* - Using the ensemble modelling technique of the gradient boosting machines(GBMs), XGBoost uses decision trees and better optimizations and enhancements to produce superior results in comparison, while using lesser computational power and showing lower execution times.

➔ K-Cross Validation Technique:

We test the effectiveness of our algorithms using the cross validation technique. In this technique, the data is divided into ‘k’ number of sets or *folds*, and one fold is considered for testing while the rest are considered for training. Multiple iterations of the training of the model are run, in which a different fold is considered for testing with each iteration. An accuracy score (MAE and RMSE in our case) is obtained for every iteration, and the arithmetic mean of the scores is considered as the overall accuracy, which usually gives the best performance accuracy. This process is illustrated as follows:

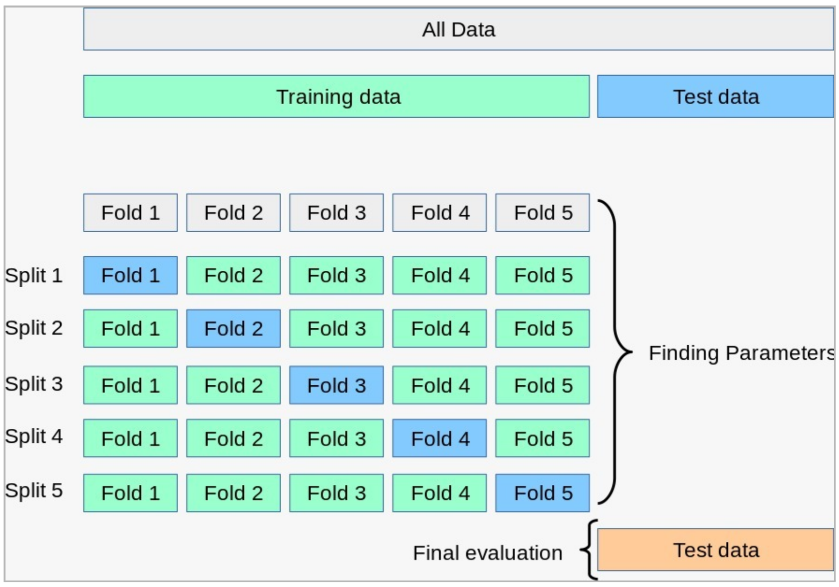


Figure 2 - Working of K-Cross Validation technique

➔ K-Means Clustering

K-Means Clustering is a type of unsupervised learning algorithm which is used with unlabeled data, by finding ‘k’ number of groups/clusters into which the data is divided, based on similar variables or factors among the data. The process occurs as follows:

- ‘K’ number of random data centroids are chosen. ‘K’ number of random data points or random variable values are chosen as starting points.

- For every data point, the centroid closest to it is located based on measures such as euclidean or cosine distance.

- The data point is then assigned to the closest centroid.

- After assigning all such points, each centroid is shifted to the average of all the data points assigned to it.

- The previous 3 steps are repeated until the centroid no longer shifts. Then the assignment of the centroid is finalized and the algorithm is said to have converged.

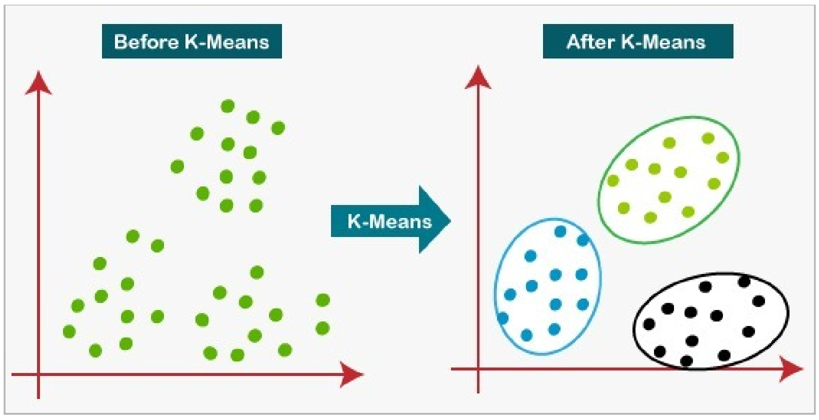


Figure 3 - Clustering process of K-Means Clustering algorithm

➔ In our case, we use K = 7 (seven clusters). The data points and cluster regions are chosen such that each cluster has similar weather conditions (Average Rainfall depth) and closer geographical coordinates. Initially, our algorithm assigns random points as centroids, then uses Euclidean distance for locating every data point. The algorithm then proceeds to classify respective clusters based on the process mentioned above. It continues iterating until there is no chance of changing the points to another cluster, and the centroid is fixed.

➔ Applying this algorithm on our dataset, we divide the country of Australia into seven different clusters or ‘zones’. Finally, the three machine learning algorithms are implemented on the data of each zone. The accuracy scores (MAE and RMSE) are obtained for the algorithms in each zone, and the best suited algorithm for each zone is determined. This also gives the best suited algorithm which can be used overall, based on the performance of the algorithm in all seven zones.

6. Schedule and Milestones

Timeline	Progress
December 1st - January 1st	Completion of Literature Survey and Analysis of Current Research and Improvements
January 1st - February 1st	Learning and Implementing Preprocessing and Feature Engineering Techniques
February 1st - March 1st	Implementation of Machine Learning Algorithms and comparison of their performance measures
March 1st - April 1st	Completion of project and design of the final Project Report
April 1st - May 1st	Final Project Review and completion of research paper thesis

7. Project Demonstration

A. Data Preprocessing and Feature Engineering:

❖ The required python libraries are imported first. Then the dataset is read and analyzed:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#import tensorflow as tf
import time
```

```
%%time
data=pd.read_csv("D:\\csv files\\weatherAUS.csv")
data.head()
```

❖ Missing values in the data are imputed via median of the non missing values:

```
def impute_nan(df,variable):
    df[variable]=df[variable].fillna(df[variable].median())
```

```
data.columns
```

```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

```
data_num_columns=['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
               'WindGustSpeed','WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
               'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
               'Temp3pm']
```

```
for x in data_num_columns:
    impute_nan(data,x)
```

❖ Missing values can also be imputed categorically by picking the most common value:

```
def cat_imputation(data,variable):
    mode=data[variable].mode()[0]
    data[variable]=data[variable].fillna(mode)
```

```
data_cat_columns=['Location','WindGustDir', 'WindDir9am', 'WindDir3pm','RainToday', 'RainTomorrow']
```

```
for x in data_cat_columns:
    cat_imputation(data,x)
```

❖ Labels of the categories are converted into sequential numbers (Label Encoding):

```
from sklearn import preprocessing
label_encoder=preprocessing.LabelEncoder()

def cat_encoding(data,variable):
    data[variable]= label_encoder.fit_transform(data[variable])
```

```
data.columns
```

```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

```
data_cat_columns=['Location','WindGustDir', 'WindDir9am', 'WindDir3pm','RainToday', 'RainTomorrow']
```

```
for x in data_cat_columns:
    cat_encoding(data,x)
```

❖ As we see below, the pressure values are on an extremely high scale compared to the rest of the variables. So we normalize all the values to fit to one smaller scale:

Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm
71.0	22.0	1007.7	1007.1	8.0	NaN	16.9	21.8
44.0	25.0	1010.6	1007.8	NaN	NaN	17.2	24.3
38.0	30.0	1007.6	1008.7	NaN	2.0	21.0	23.2
45.0	16.0	1017.6	1012.8	NaN	NaN	18.1	26.5
82.0	33.0	1010.8	1006.0	7.0	8.0	17.8	29.7

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
data['Pressure9am']=scaler.fit_transform(data[['Pressure9am']])
data['Pressure3pm']=scaler.fit_transform(data[['Pressure3pm']])
```

❖ We observe the heatmap below of the correlation between the different variables in the dataset.

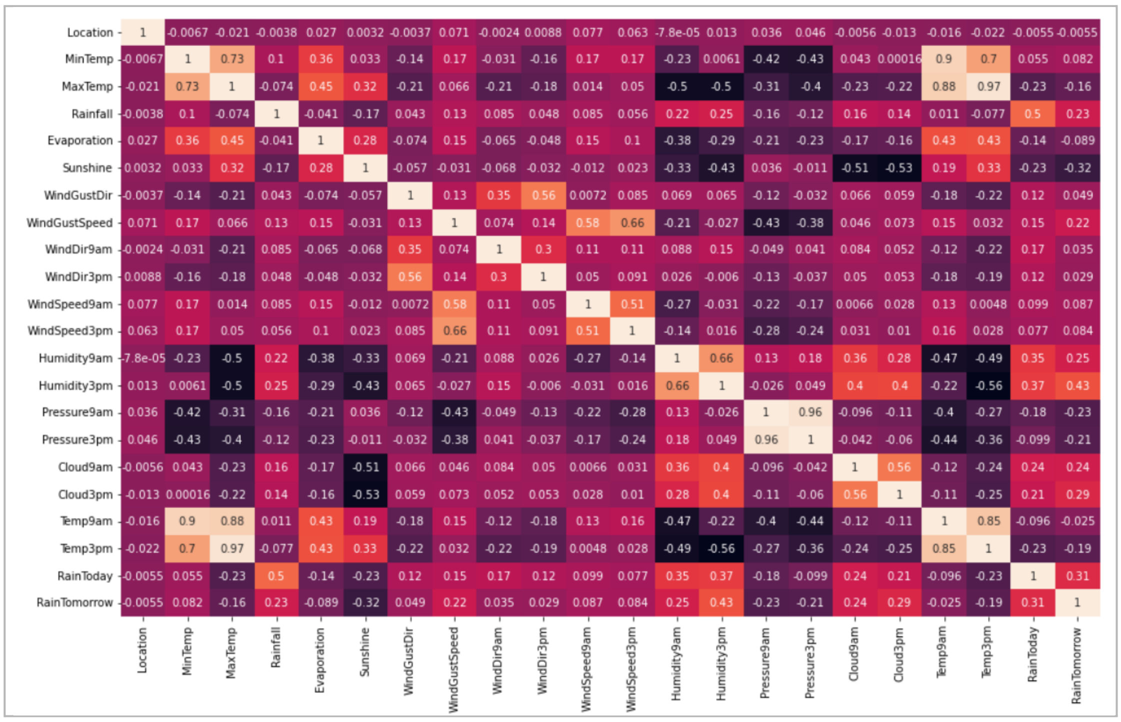


Figure 4 - Heatmap visualization of the degree of correlation among variables in data

❖ We notice that the variables 'Temp9am', 'Temp3pm', 'MaxTemp', and 'MinTemp' are highly correlated to each other which leads to a Multicollinearity issue. We can compare their VIF values to understand the amount of correlation:

Table 1 - Variables in the data and their corresponding Variable Inflation Factors (VIF's)

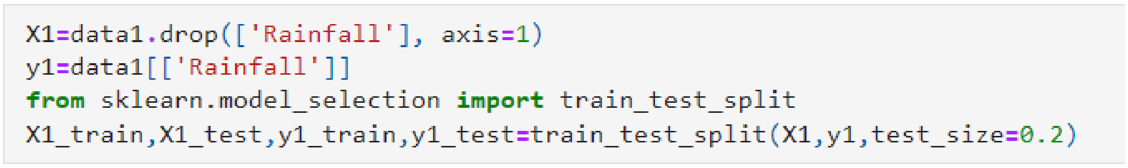
Variable	**VIF**	**Variable**	**VIF**
Location	3.851119	Humidity9am	48.392861
MinTemp	36.137234	Humidity3pm	36.312532
MaxTemp	250.858306	Pressure9am	602.781636
Evaporation	5.020835	Pressure3pm	621.090430
Sunshine	15.229978	Cloud9am	8.832513
WindGustDir	6.320029	Cloud3pm	10.248982
WindGustSpeed	22.985536	Temp9am	141.815752
WindDir9am	4.297737	Temp3pm	237.317249
WindDir3pm	6.295325	RainToday	1.761341
WindSpeed9am	6.332815	RainTomorrow	1.859505
WindSpeed3pm	10.920684		

❖ We remove the four highly correlated variables, and the two pressure variables which are scaled very high in the original data. This creates a more uniform and balanced data:

Table 2 - Variables and their VIF's after resolving multicollinearity issue

Variable	**VIF**	**Variable**	**VIF**
Location	3.679918	WindSpeed9am	4.949934
MinTemp	5.741403	WindSpeed3pm	7.750609
Evaporation	4.848245	Humidity3pm	10.311760
Sunshine	7.816628	Cloud9am	8.091534
WindGustDir	6.167268	Cloud3pm	9.122193
WindDir9am	4.133418	RainToday	1.590356
WindDir3pm	5.967294	RainTomorrow	1.704695

❖ Now that our data is shaped uniformly, it is ready to be used to train the machine learning algorithms. We convert the data into training and testing sets, with 80% training and 20% testing division:

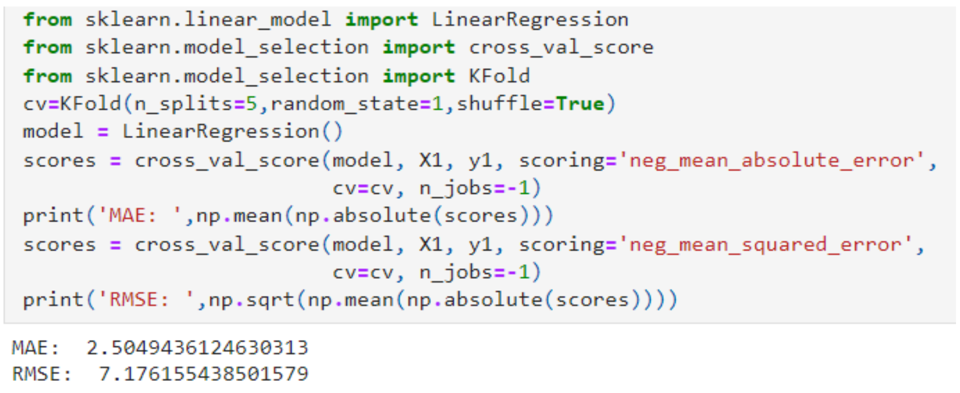


```
X1=data1.drop(['Rainfall'], axis=1)
y1=data1[['Rainfall']]
from sklearn.model_selection import train_test_split
X1_train,X1_test,y1_train,y1_test=train_test_split(X1,y1,test_size=0.2)
```

B. <u>Implementing the Machine Learning Models</u>:

- First we import and train the machine learning models using our training and testing sets. We implement the k-cross validation algorithm to perform different iterations using different testing sets for each iteration.

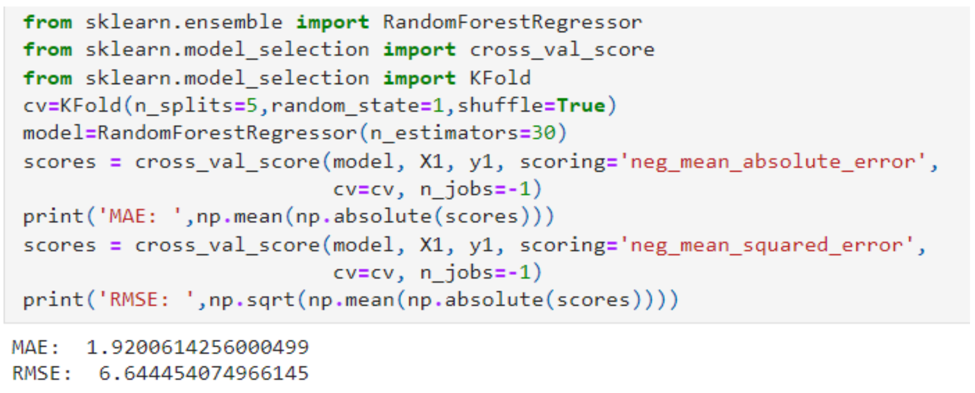
- First, we train the *Linear Regression Algorithm*:



```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
cv=KFold(n_splits=5,random_state=1,shuffle=True)
model = LinearRegression()
scores = cross_val_score(model, X1, y1, scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)
print('MAE: ',np.mean(np.absolute(scores)))
scores = cross_val_score(model, X1, y1, scoring='neg_mean_squared_error',
                         cv=cv, n_jobs=-1)
print('RMSE: ',np.sqrt(np.mean(np.absolute(scores))))
```

```
MAE:  2.5049436124630313
RMSE:  7.176155438501579
```

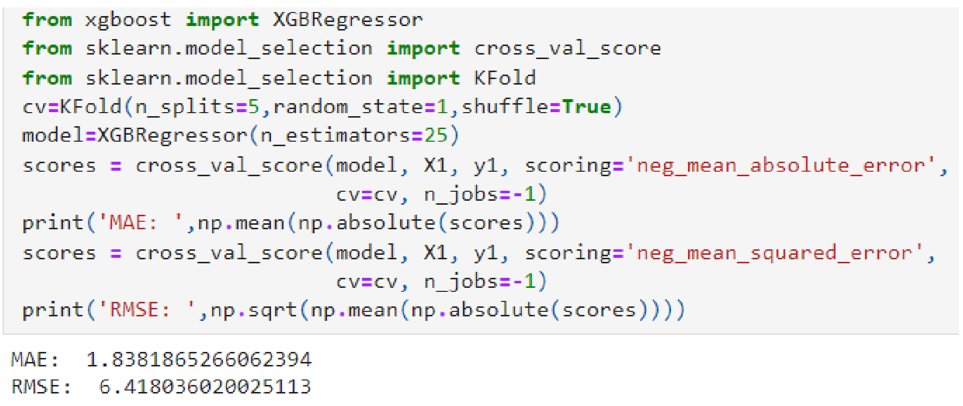
- Next, we train the *Random Forest Algorithm*:



```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
cv=KFold(n_splits=5,random_state=1,shuffle=True)
model=RandomForestRegressor(n_estimators=30)
scores = cross_val_score(model, X1, y1, scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)
print('MAE: ',np.mean(np.absolute(scores)))
scores = cross_val_score(model, X1, y1, scoring='neg_mean_squared_error',
                         cv=cv, n_jobs=-1)
print('RMSE: ',np.sqrt(np.mean(np.absolute(scores))))
```

```
MAE:  1.9200614256000499
RMSE:  6.644454074966145
```

- Then, the *XGBoost Algorithm*:



```
from xgboost import XGBRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
cv=KFold(n_splits=5,random_state=1,shuffle=True)
model=XGBRegressor(n_estimators=25)
scores = cross_val_score(model, X1, y1, scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)
print('MAE: ',np.mean(np.absolute(scores)))
scores = cross_val_score(model, X1, y1, scoring='neg_mean_squared_error',
                         cv=cv, n_jobs=-1)
print('RMSE: ',np.sqrt(np.mean(np.absolute(scores))))
```

```
MAE:  1.8381865266062394
RMSE:  6.418036020025113
```

- Hence we obtain the performance scores of each algorithm when implemented on our data. We carry out a complete performance analysis and comparison of the three algorithms in the upcoming sections

C. <u>K-Means Clustering and Cluster Analysis</u>

- We use Tableau software to help in the visualization of the clustering process. Tableau is also used here to generate the geographical coordinates of each location which we use later to form the clusters. We can observe the various locations in the country *before* the clustering, as follows:

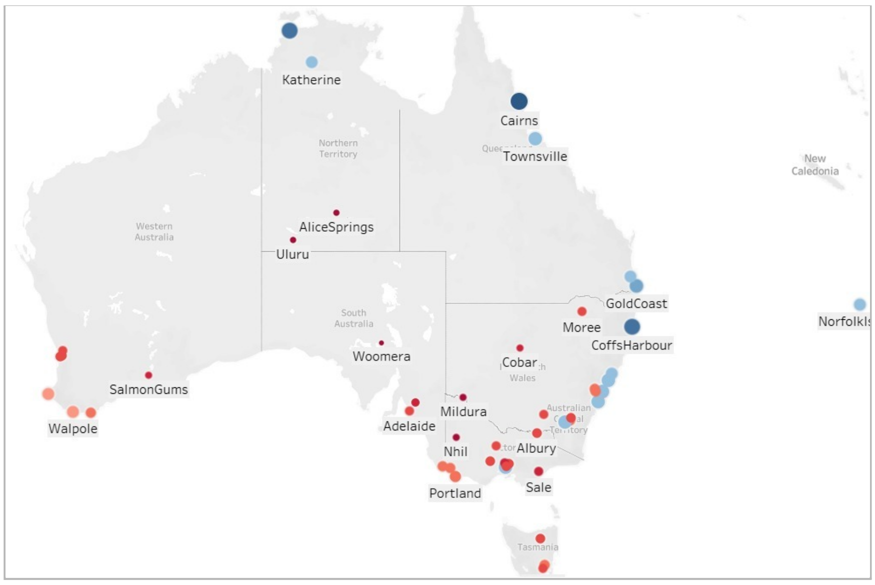


Figure 5 - Tableau map display of the various geographical locations in our data

- The bubble chart below indicates the average amount of rainfall in each location, with the size of the bubble increasing proportionally to the amount of rainfall:

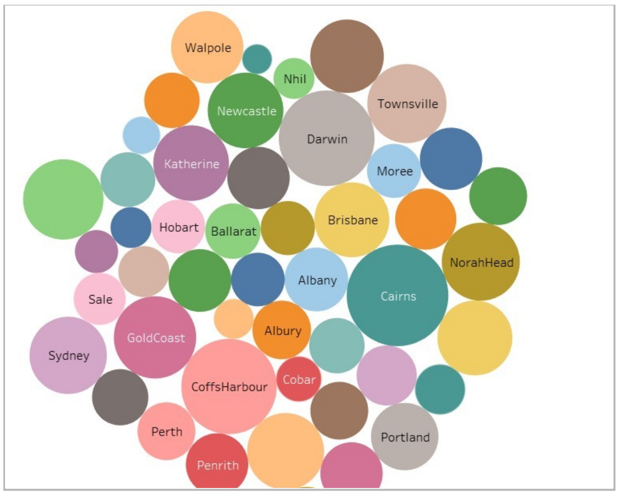


Figure 6 - Bubble chart depicting the average rainfall of each location

❖ We can clearly observe the split of the seven zones from the tableau graph, which have been clustered based on geographical location. Each color represents a specific zone:

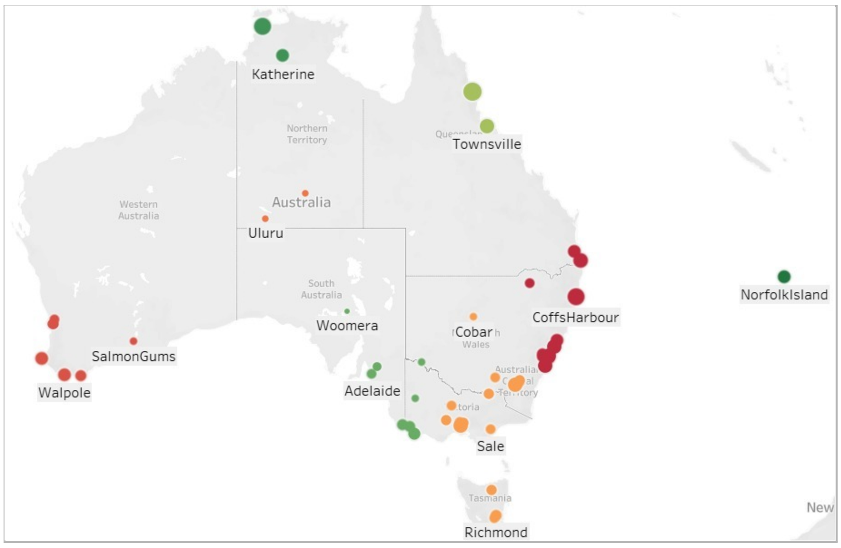
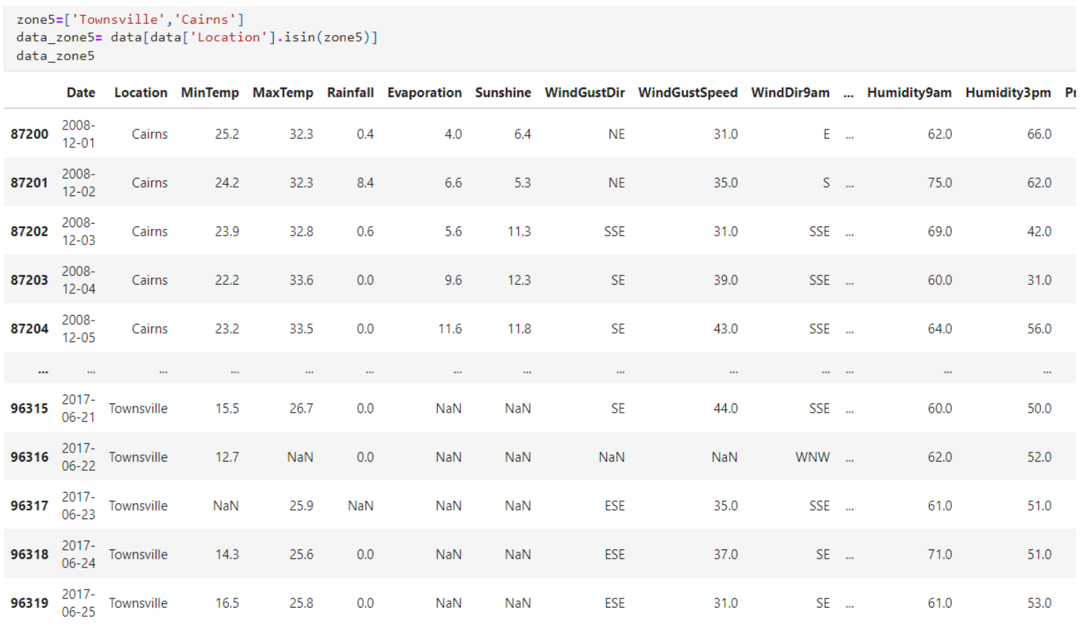


Figure 7 - Tableau map visualization of the various clustered zones shown in different colors

❖ For example, the locations 'Cairns' and 'Townsville' have been fit into a single zone due to their closeness in geographical distance. In this case they have been classified as 'zone 5'.

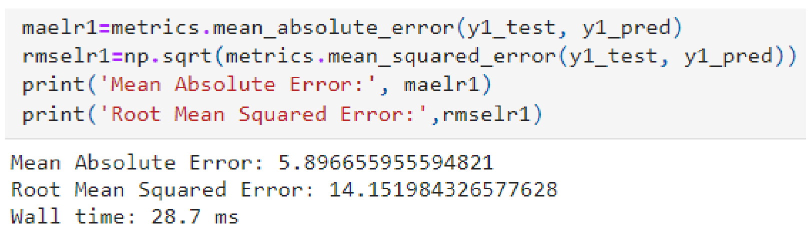


```
zone5=['Townsville','Cairns']
data_zone5= data[data['Location'].isin(zone5)]
data_zone5
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pr
87200	2008-12-01	Cairns	25.2	32.3	0.4	4.0	6.4	NE	31.0	E	...	62.0	66.0	
87201	2008-12-02	Cairns	24.2	32.3	8.4	6.6	5.3	NE	35.0	S	...	75.0	62.0	
87202	2008-12-03	Cairns	23.9	32.8	0.6	5.6	11.3	SSE	31.0	SSE	...	69.0	42.0	
87203	2008-12-04	Cairns	22.2	33.6	0.0	9.6	12.3	SE	39.0	SSE	...	60.0	31.0	
87204	2008-12-05	Cairns	23.2	33.5	0.0	11.6	11.8	SE	43.0	SSE	...	64.0	56.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
96315	2017-06-21	Townsville	15.5	26.7	0.0	NaN	NaN	SE	44.0	SSE	...	60.0	50.0	
96316	2017-06-22	Townsville	12.7	NaN	0.0	NaN	NaN	NaN	NaN	WNW	...	62.0	52.0	
96317	2017-06-23	Townsville	NaN	25.9	NaN	NaN	NaN	ESE	35.0	SSE	...	61.0	51.0	
96318	2017-06-24	Townsville	14.3	25.6	0.0	NaN	NaN	ESE	37.0	SE	...	71.0	51.0	
96319	2017-06-25	Townsville	16.5	25.8	0.0	NaN	NaN	ESE	31.0	SE	...	61.0	53.0	

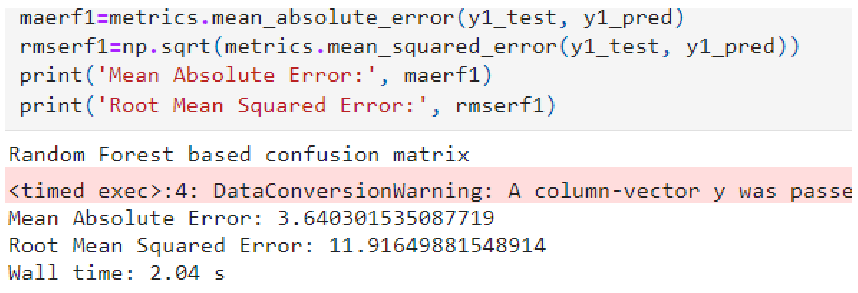
❖ Now, we implement the Feature Engineering and Machine Learning Algorithms on each of the seven zones, and obtain the accuracy scores (MAE and RMSE) for the performance in each zone, and finally the best suited algorithm for each zone is determined. This is then used to conclude the best suited algorithm which can be used overall, based on the performance of it in each of the seven zones.

For example, shown below are the outputs of the ***linear regression, random forest and xgboost algorithm respectively***, for the previously observed zone, 'zone 5'.



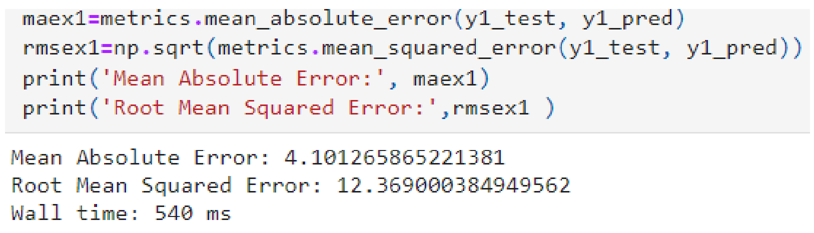
```
maelr1=metrics.mean_absolute_error(y1_test, y1_pred)
rmselr1=np.sqrt(metrics.mean_squared_error(y1_test, y1_pred))
print('Mean Absolute Error:', maelr1)
print('Root Mean Squared Error:',rmselr1)
```

```
Mean Absolute Error: 5.896655955594821
Root Mean Squared Error: 14.151984326577628
Wall time: 28.7 ms
```



```
maerf1=metrics.mean_absolute_error(y1_test, y1_pred)
rmserf1=np.sqrt(metrics.mean_squared_error(y1_test, y1_pred))
print('Mean Absolute Error:', maerf1)
print('Root Mean Squared Error:', rmserf1)
```

```
Random Forest based confusion matrix
<timed exec>:4: DataConversionWarning: A column-vector y was passe
Mean Absolute Error: 3.640301535087719
Root Mean Squared Error: 11.91649881548914
Wall time: 2.04 s
```



```
maex1=metrics.mean_absolute_error(y1_test, y1_pred)
rmsex1=np.sqrt(metrics.mean_squared_error(y1_test, y1_pred))
print('Mean Absolute Error:', maex1)
print('Root Mean Squared Error:',rmsex1 )
```

```
Mean Absolute Error: 4.101265865221381
Root Mean Squared Error: 12.369000384949562
Wall time: 540 ms
```

❖ We present the detailed performance comparison of the three algorithms in each zone in the upcoming section.

8. Results and Discussion

A. Performance Analysis and Comparison (Overall):

➢ We observe the performance of the three algorithms on the complete processed data shown below. The table shows a comparison of the Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and the CPU and GPU Execution Times of each of the algorithms:

Table 3 - Performances and comparison of various algorithms used

Algorithm	MAE	RMSE	CPU Time	GPU Time
Linear Regression	2.485	7.239	373 ms	171 ms
Random Forest	1.872	6.589	43.5 secs	21.5 secs
XGBoost	1.787	6.393	4.5 secs	3.27 secs

➢ We can clearly see that the ***XGBoost*** algorithm exceeds the others in terms of minimal error values, while also having reasonable execution times without the necessity of high computational power. Hence we can conclude that the XGBoost model is best suited for prediction in this case.

B. Performance Comparison with Base Reference Paper:

➢ We display below the values obtained from each of the three algorithms in our base reference paper, and a comparison of them with the values obtained in our work:

Table 4 - Performance comparison of our design vs.base reference paper

	MAE		RMSE	
Algorithm	Base Paper	Our Design	Base Paper	Our Design
Linear Regression	4.97	2.485	8.61	7.239
Random Forest	4.49	1.872	8.82	6.589
XGBoost	3.58	1.787	7.85	6.393

\

- We can clearly see that our work has provided significantly lower error values in comparison to our referred base paper, thus showing an improvement in reliability of prediction.

C. Cluster Analysis and Zone-Wise Performance:

- Now we compare our obtained accuracy scores from each of the seven zones, and show the algorithm best suited for each zone:

Table 5-12 - Performance comparison of algorithms in each of the 7 zones

ZONE 1		
Algorithm	**MAE**	**RMSE**
LR	3.36	9.16
RF	2.84	9.32
XGBoost	2.71	9.08

ZONE 2		
Algorithm	**MAE**	**RMSE**
LR	1.74	4.34
RF	1.52	4.35
XGBoost	1.56	4.37

ZONE 3		
Algorithm	**MAE**	**RMSE**
LR	1.06	3.52
RF	0.73	3.67
XGBoost	0.77	3.87

ZONE 4		
Algorithm	**MAE**	**RMSE**
LR	1.87	5.30
RF	1.57	5.21
XGBoost	1.53	5.16

ZONE 5		
Algorithm	**MAE**	**RMSE**
LR	6.15	14.21
RF	3.73	12.3
XGBoost	3.95	12.23

ZONE 6		
Algorithm	**MAE**	**RMSE**
LR	1.41	3.71
RF	1.13	3.62
XGBoost	1.09	3.54

ZONE 7		
Algorithm	**MAE**	**RMSE**
LR	5.09	12.41
RF	3.01	9.93
XGBoost	2.85	10.21

OVERALL DATA		
Algorithm	**MAE**	**RMSE**
LR	**2.5**	**7.17**
RF	**1.92**	**6.64**
XGBoost	**1.83**	**6.41**

- It is clear that ***XGBoost*** is the most reliable algorithm based on the zonal performances of the three algorithms. The model shows low error values in most of the cases, and is very close behind in the few cases where other models excel.

- Therefore, through observation of all the cases of performance analysis, it is reasonable to conclude that the XGBoost algorithm is the most effective algorithm to use.

9. Summary

Our design for the prediction of rainfall has provided optimal results under most circumstances. It has shown great improvement in accuracy scores in comparison to our reference paper. The analysis of clusters has also shown how our algorithm can perform when provided with different types of data in different distributions. It has proven to be more time efficient, highly reliable, and simple to understand and execute, and is therefore ready to be used in real time to predict rainfall in a location.

10. References

- ☐ *Liyew, Chalachew & Melese, Haileyesus. (2021). Machine learning techniques to predict daily rainfall amount. Journal of Big Data. 8. 10.1186/s40537-021-00545-4.*
- ☐ *Atitallah, S.B., et al.: Leveraging Deep Learning and IoT big data analytics to support the smart cities development: review and future directions. Comput. Sci. Rev. 38, 100303 (2020)*
- ☐ *Dai, Weijun & Tang, Yanni & Zhang, Zeyu & Cai, Zhiming. (2021). Ensemble Learning Technology for Coastal Flood Forecasting in Internet-of-Things-Enabled Smart City. International Journal of Computational Intelligence Systems.14. 166. 10.1007/s44196-021-00023-y.*
- ☐ *B. M. Shankar, T. J. John, S. Karthick, B. Pattanaik, M. Pattnaik and S. Karthikeyan, "Internet of Things based Smart Flood forecasting and Early Warning System," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 443-447, doi: 10.1109/ICCMC51019.2021.9418331.*
- ☐ *Moussa, Moustafa & Zhang, Xiangliang & Claudel, Christian. (2016). Flash Flood Detection in Urban Cities Using Ultrasonic and Infrared Sensors. IEEE Sensors Journal. 16. 1-1. 10.1109/JSEN.2016.2592359.*
- ☐ *Namitha K, Jayapriya A, Santhosh Kumar G. Rainfall prediction using artificial neural network on map-reduce framework. ACM. 2015.*
- ☐ *Tharun VP, Prakash R, Devi SR. Prediction of Rainfall Using Data Mining Techniques. In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT). IEEE Xplore. 2018; pp. 1*
- ☐ *Garg, Arnav & Pandey, Himanshu. (2019). Rainfall Prediction Using Machine Learning. 10.13140/RG.2.2.26691.04648.*
- ☐ *Aswin S, Geetha P, Vinayakumar R. Deep learning models for the prediction of rainfall. In 2018 International Conference on Communication and Signal Processing (ICCSP). IEEE: New York. 2018; pp. 0657*

11. Appendix

a. Mathematical Background

❖ ***Mean Absolute Error (MAE):***

Absolute error is defined as the total amount of error in a measurement, and it is the absolute difference between the measured value and the 'true' value of a variable. The equation for calculation of the absolute error is defined as follows:

$$(\Delta x) = |xi - x|$$

Where,
Δx is the absolute error.
xi is the measurement, and
x is the true value.

Mean Absolute Error is the arithmetic mean of all of the absolute errors. The mathematical equation for calculating MAE is shown as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$

Where,
Σ is the summation, and
n is the total sample size.

❖ ***Root Mean Square Deviation (RMSD):***

Root Mean Square Error or *Root Mean Square Deviation (RMSD)* shows how far away our predicted values are from the true values. The mathematical equation for calculating RMSD is shown as follows:

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}$$

Where,
Σ is the summation,
√ is the square root,
$\hat{x}_i$ is the measurement,
x_i is the true value, and
N is the total sample size.

b. Algorithms

❖ ***Linear Regression Algorithm:***

Linear Regression is a machine learning algorithm based on *supervised learning*. It performs a regression task, which is to map an independent variable (x) in the data in a linear way to our dependent variable which is to be predicted (y).

The linear regression model provides a sloped straight line representing the relationship between the variables:

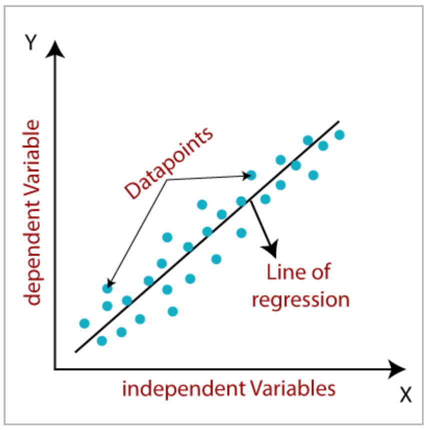


Figure A1 - Illustration of Linear Regression model

Mathematically, the equation for linear regression can be expressed as follows:

$$\mathbf{y = a_0 + a_1 x + \varepsilon}$$

Where,
y is the dependent variable (target variable)
x is the independent variable (predictor variable)
a_0 is the intercept of the line (Gives an additional degree of freedom)
a_1 is the linear regression coefficient (scale factor to each input value).
ε is the random error

Linear Regression can be *Simple* or *Multiple*, based on whether a single independent variable, or multiple independent variables are used to predict the dependent variable.

❖ *Random Forest Algorithm:*

Random Forest is a machine learning algorithm also based on supervised learning which can be used for both Classification and Regression problems. It is based on ensemble learning, which is a method that uses multiple learning algorithms to obtain better predictive performance than what could be obtained from any single one of the constituent learning algorithms.

The Random Forest classifier consists of multiple *decision trees* imposed on various subsets of the given dataset and takes the average of all of them to improve the predictive accuracy of that dataset.

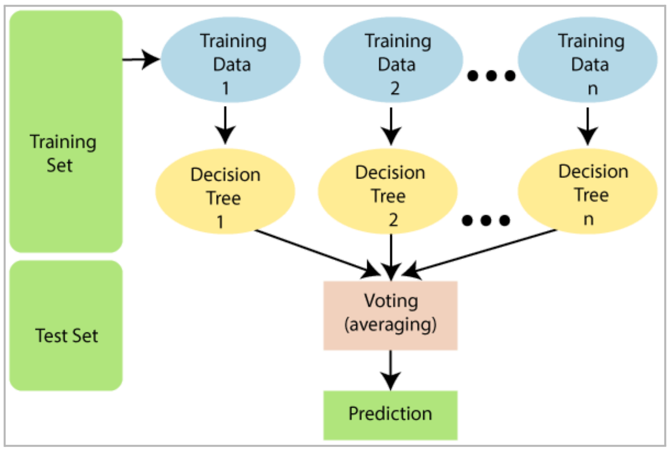


Figure A2 - Illustration of Random Forest model

There are mainly two types of methods used in ensemble learning:

➔ *Bagging* - Also known as Bootstrap Aggregation, this is the technique used by the random forest algorithm. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement, known as row sampling. Then, each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance

➔ *Boosting* - Boosting is a technique that attempts to build a strong classifier from the number of weak classifiers used in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added. This technique is used in gradient boosting which is then implemented in the XGBoost algorithm.

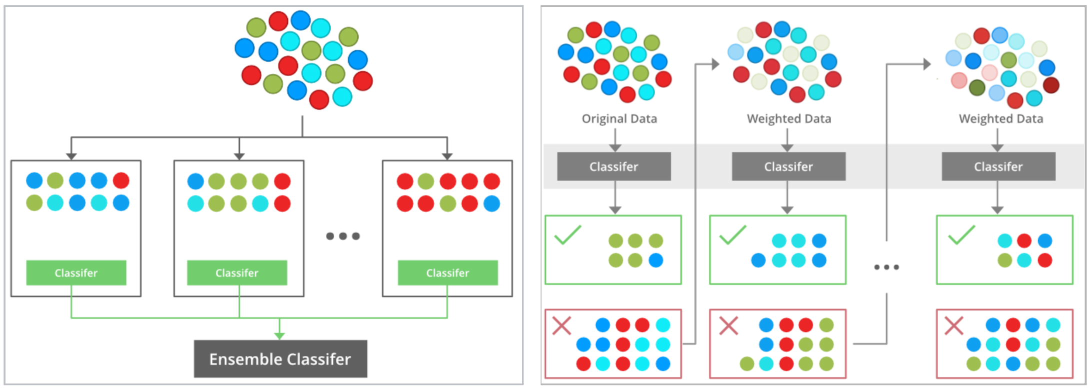


Figure A3 - Comparison of Bagging (left) vs Boosting (right) ensemble learning methods

❖ ***XGBoost Algorithm:***

XGBoost is an implementation of *Gradient Boosted decision trees*. In gradient boosting, each predictor corrects its predecessor's error, and is trained using the residual errors of the predecessor as labels.

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.