PREDICTIVE ANALYSIS ON RAINFALL STATISTICS IN MAHARASHTRA

A course project in the domain of predictive analytics for the course of Big Data Analytics (IC3026)

UNDERTAKEN BY

ANAYA PAWAR

(T. Y. Q 5, G. R. 1710263)

ANEESH PODUVAL

(T. Y. Q 6, G. R. 1710045)

SARTHAK CHUDGAR

(T. Y. Q 16, G. R. 1710202)

JOHNATHAN FERNANDES

(T. Y. Q 37, G. R. 1710168)

UNDER THE GUIDANCE OF **PROF. (DR.) KULKARNI JAYANT V. & PROF. PRIYANKA MEHTA**

TABLE OF CONTENTS

PREFACE

Rainfall plays an integral role in the lives of millions of people worldwide. This is especially true for an **agriculture heavy** country such as India. Not only do we depend on rain as a source of fresh water, we also use it to irrigate farms and for rainwater harvesting.

Given our dependence on rain, prior knowledge about the **weather forecast** is essential for optimal rainwater utilization. This is where **predictive analytics** comes in.

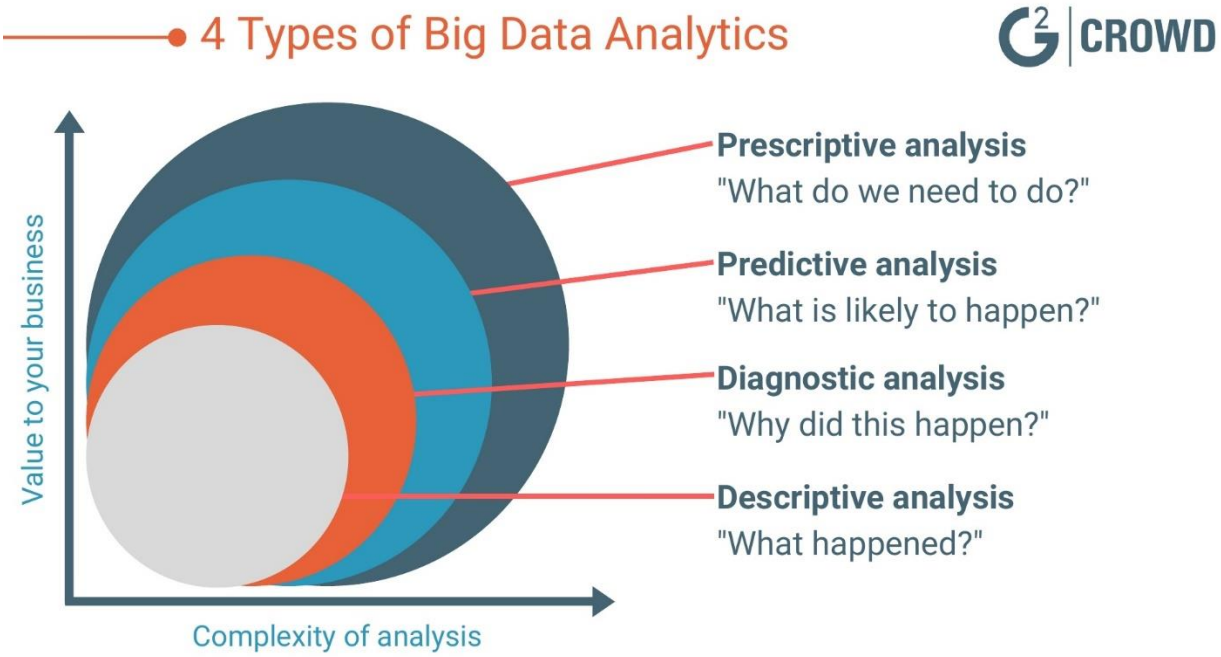

Figure 1: Types of Data Analytics

Predictive analytics is the 3rd step in big data analytics. It involves the previous two steps, i.e. **descriptive** and **diagnostic analytics**, and aims to build a model which can analyze data trends and **predict future trends** in order to help individuals better prepare themselves.

In this project we will carry out predictive analytics on weather data over the state of Maharashtra to construct a predictive model which will be able to **predict** the amount of rainfall in millimeters.

While the primary focus of this project is in the field of agriculture, it also plays a vital role in other fields such as disaster management

CASE STUDY

In order to find an effective **error margin**, we decided to conduct a **case study** on the Australian Government Bureau of Meteorology.

Figure 2: Weather forecast, showing only chance of precipitation

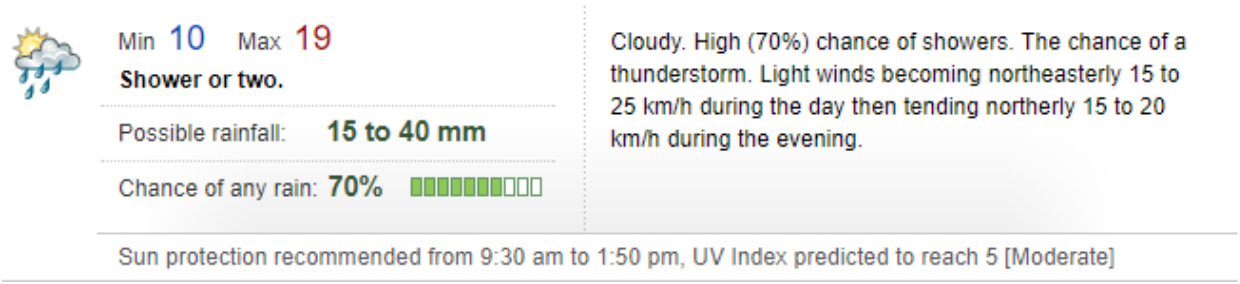

Figure 3: Weather forecast with a margin of ± 12.5mm

We noted that as far as rainfall is concerned, they do not provide an amount of rainfall, but usually a chance of precipitation instead. When a measurement is provided, it is often large range.

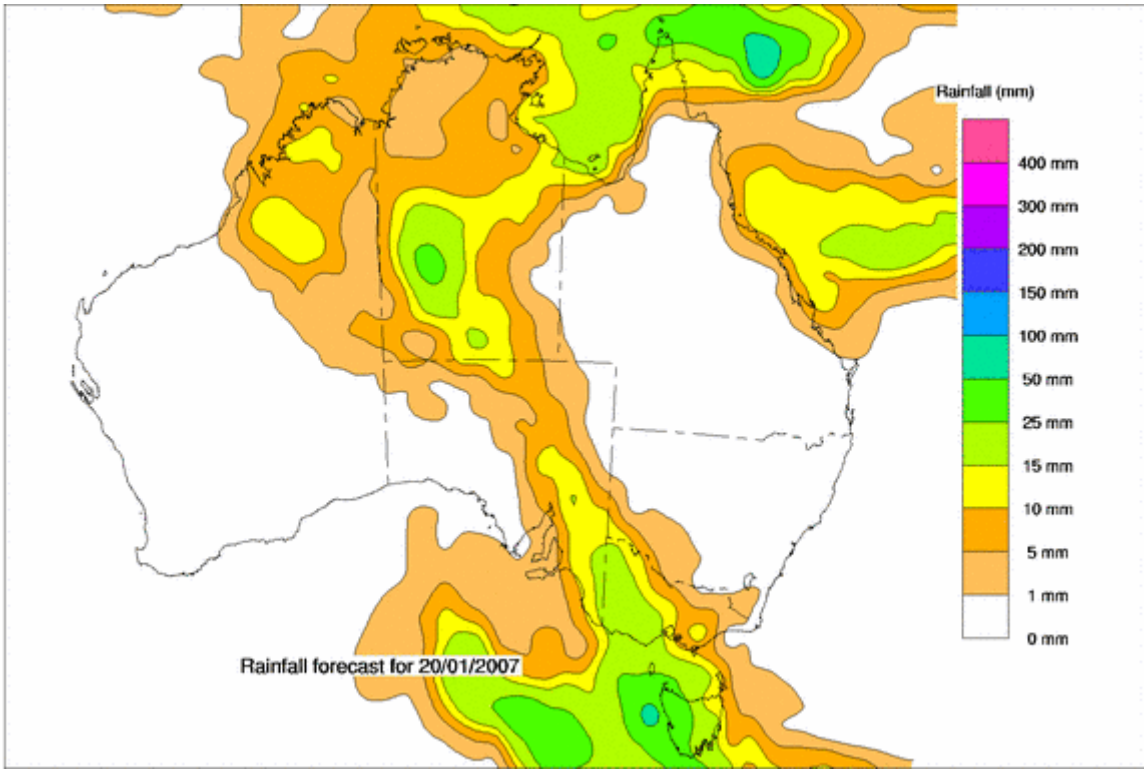

Figure 4: Predicted Rainfall for 20 January 2007

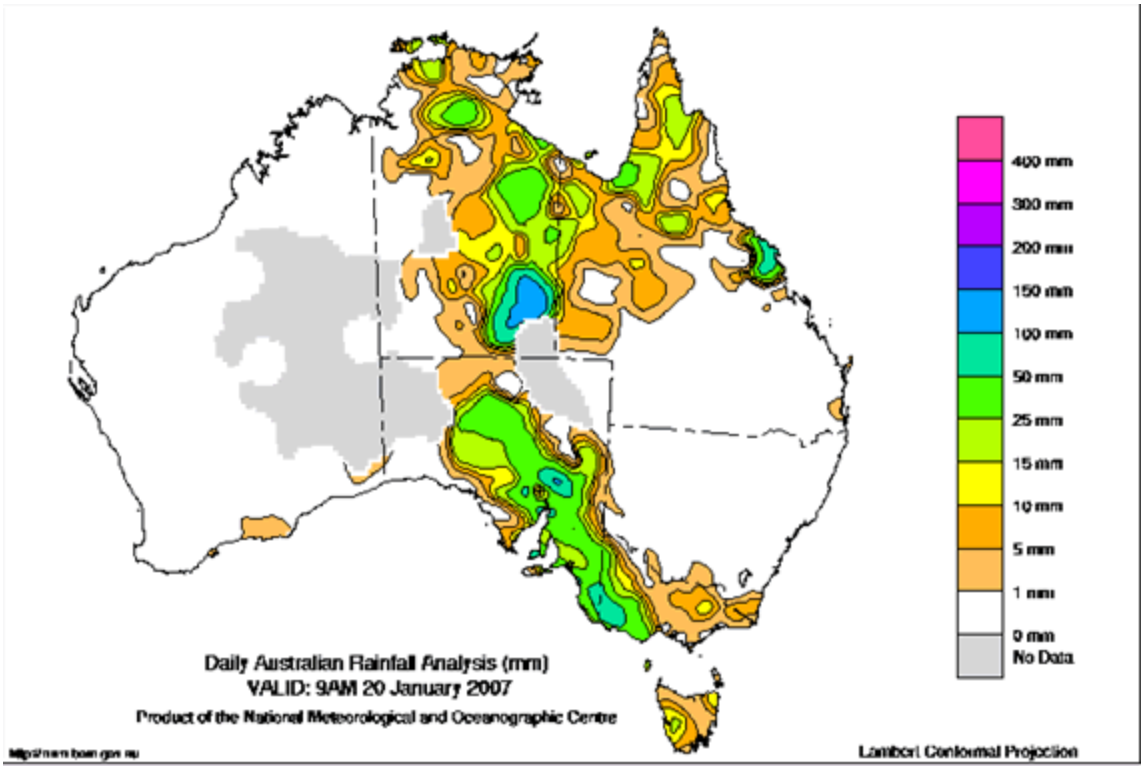

Figure 5: Actual rainfall for 20 January 2007

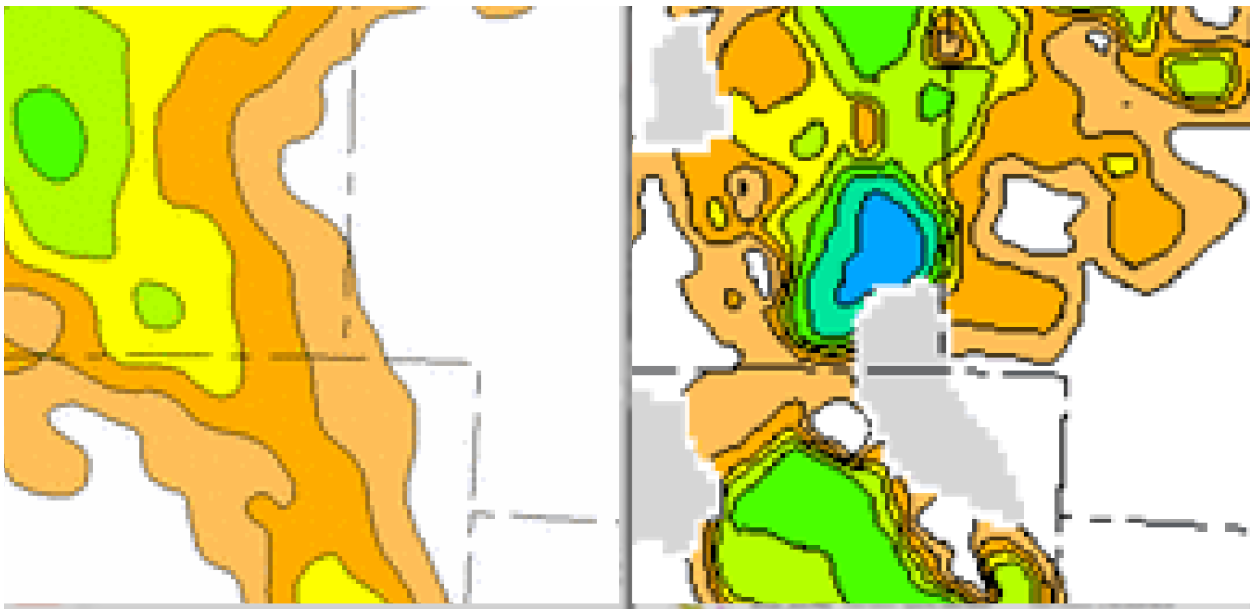

Figure 6: Side-by-side comparison of the predicted and actual forecasts

When examining rainfall prediction maps, we observe that the predictions in some areas might differ by as much as **140mm**

Keeping these observations in mind, we proceed with our analysis.

STEP 1: DESCRIPTIVE ANALYTICS

This step involves **obtaining data** of our variable (rainfall, in this case) and **possible related factors** (weather properties that may or may not affect rainfall).

We then **visualize** our dependent variable (rainfall) with each of the independent variables (other factors).

DATA ACQUISITION

We obtain our data from the National Centers for Environmental Prediction (NCEP) website. They have constructed a Climate Forecast System Reanalysis (CFSR) which provides worldwide weather readings from as far back as 1980

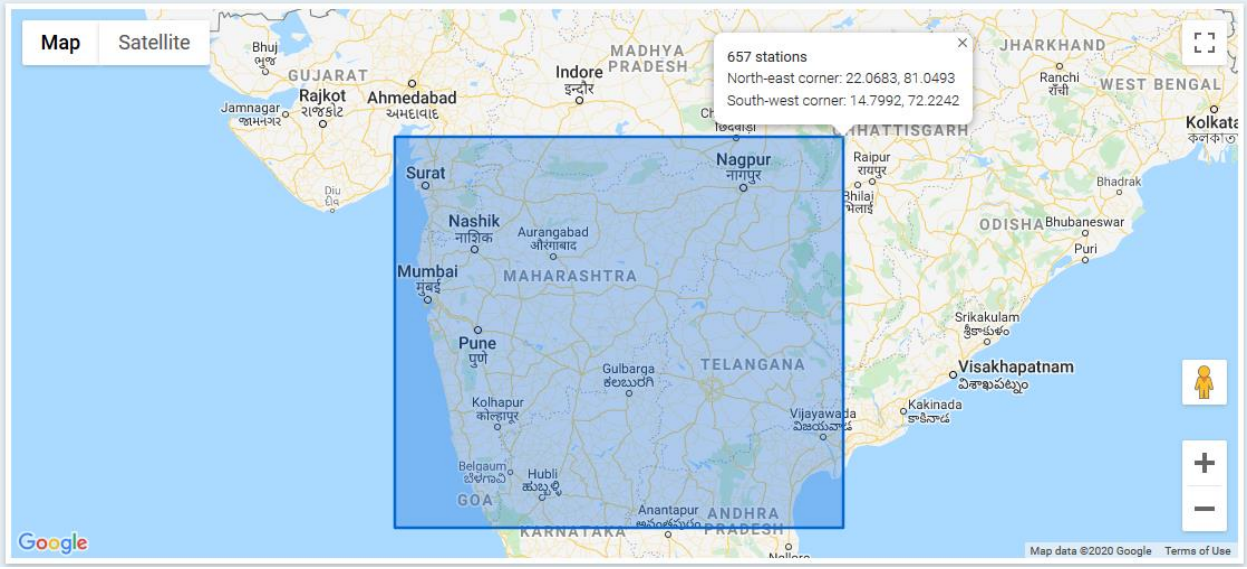

Figure 7: NCEP Map GUI

The NCEP website provides a **map GUI** to choose an area, and then provides all data from that area. While the selection includes 657 stations, Maharashtra only contains about **370** of them. We use **Tableau** to extract those stations.

From the system, we obtain data on the following:

Weather Factor	Units & Remarks
Precipitation	Millimeters
Humidity	Fraction
Location	Latitude and Longitude
Temperature	°C, (Minimum and Maximum)
Wind speed	Meters per second
Solar Coverage	Mega joules per square meter

Table 1: Selected features

This data was taken over the period of **1980 to 2011.**

STEP 2: DIAGNOSTIC ANALYTICS

Step 2 is closely related to step 1 to the point where they are usually carried out simultaneously. It entails using the previously created visualizations to determine the **relations** between our independent and dependent variable.

We use **Tableau**, a well-known business insights tool to visualize the data into various easy to understand charts.

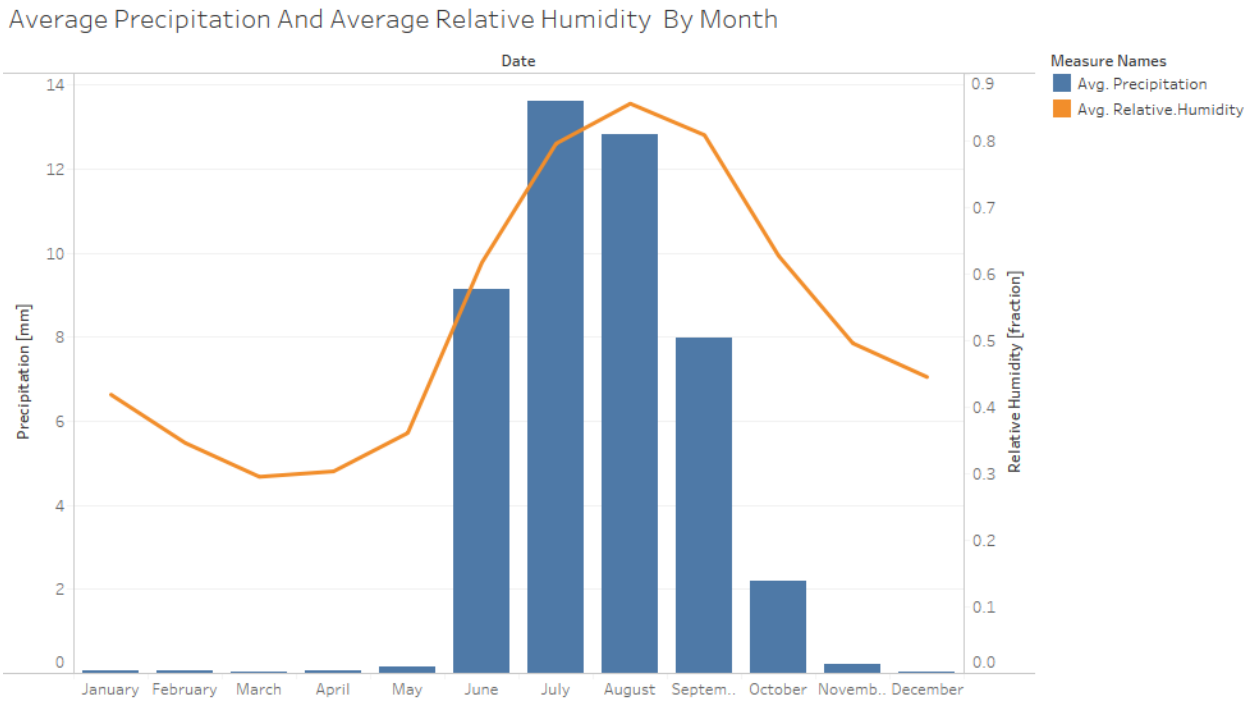

Figure 8: Average Precipitation and Average Humidity by Month

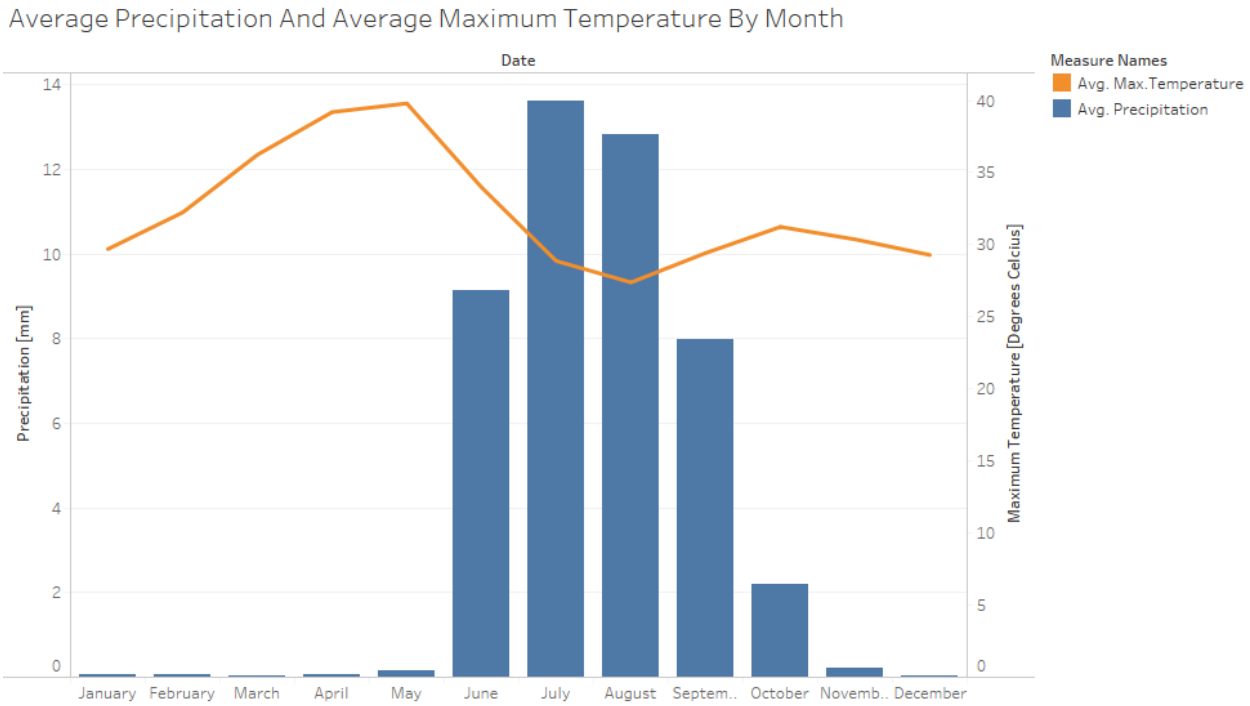

Figure 9: Average Precipitation and Average Maximum Temperature by Month

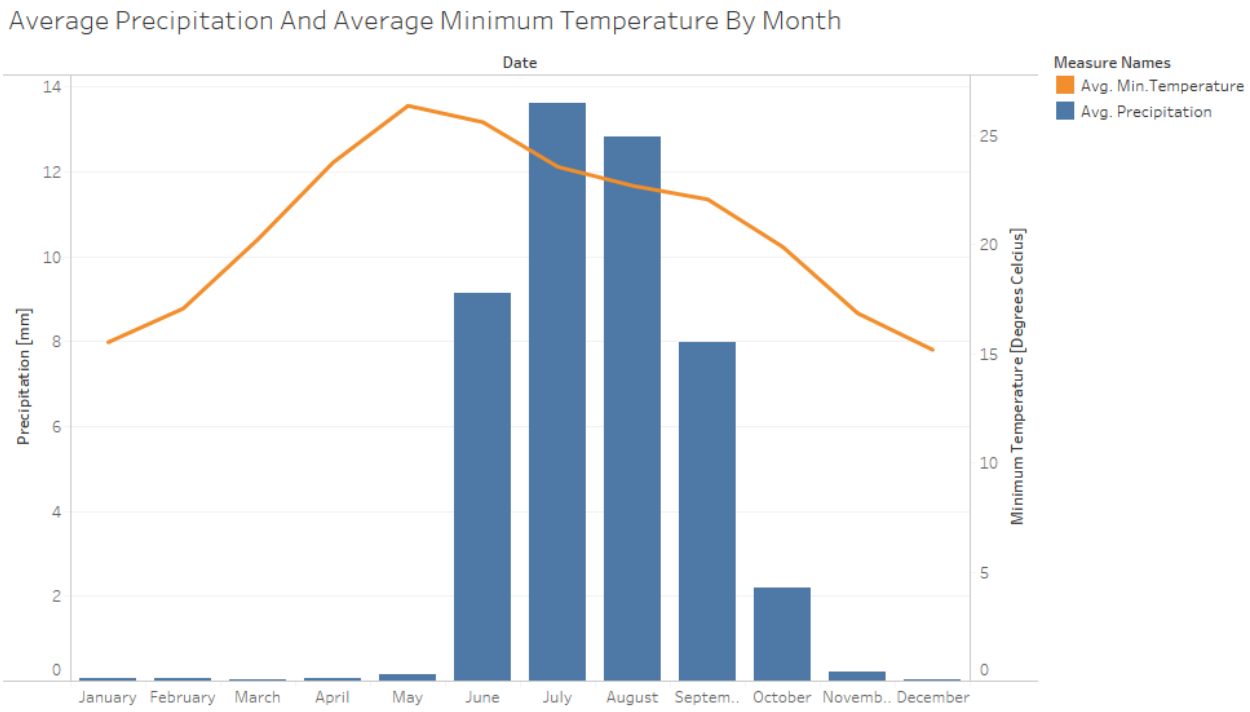

Figure 10: Average Precipitation and Average Minimum Temperature by Month

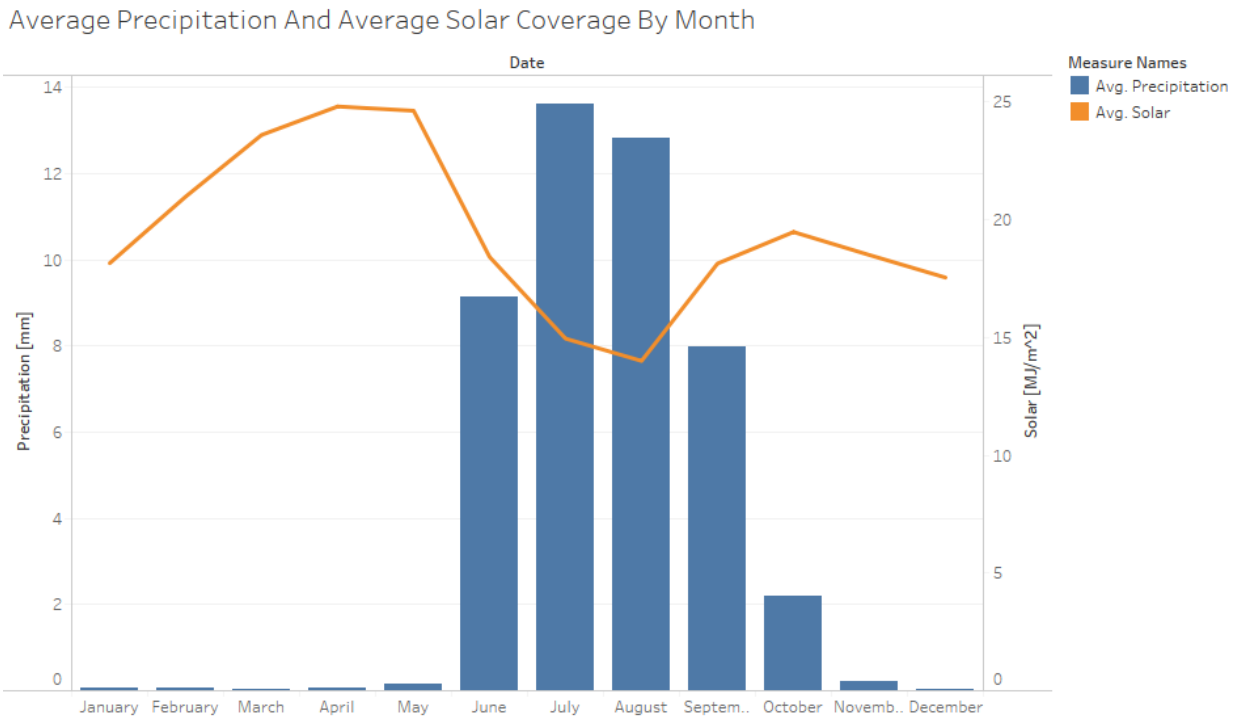

Figure 11: Average Precipitation and Average Solar Coverage by Month

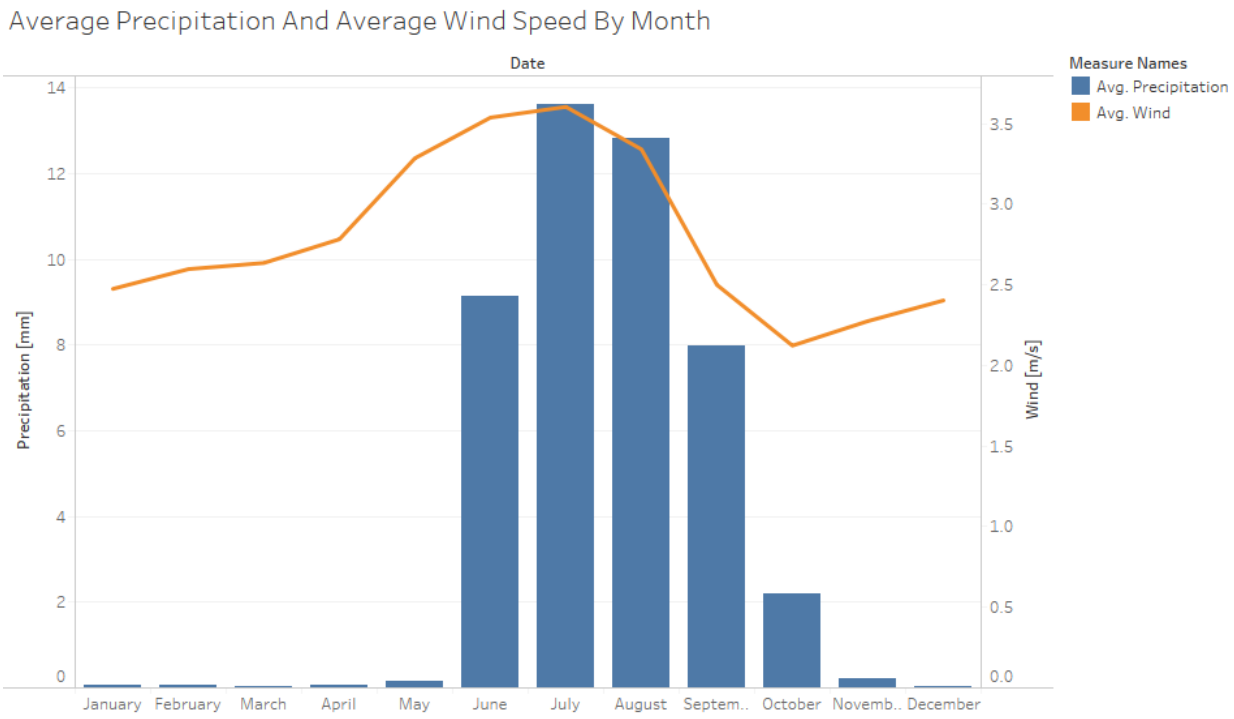

Figure 12: Average Precipitation and Average Wind Speed by Month

Upon observing the visualizations, it is clear that all the independent factors considered influence our dependent factor by a significant amount. We also cross check this using probability values as shown below

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        51.981802   1.016792   51.12   <2e-16 ***
t2$Latitude        -0.364066   0.017488  -20.82   <2e-16 ***
t2$Longitude       -0.962631   0.013580  -70.88   <2e-16 ***
t2$Max.Temperature  0.843547   0.005467  154.30   <2e-16 ***
t2$Min.Temperature -0.275606   0.004477  -61.55   <2e-16 ***
t2$Wind             3.002539   0.010101  297.24   <2e-16 ***
t2$Relative.Humidity 28.457280   0.106866  266.29   <2e-16 ***
t2$Solar           -0.784839   0.002709 -289.68   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.15 on 1048567 degrees of freedom
Multiple R-squared:  0.3054,    Adjusted R-squared:  0.3054
F-statistic: 6.585e+04 on 7 and 1048567 DF,  p-value: < 2.2e-16
```

Figure 13: Probability values of linear model

We initially constructed a simple linear regression model to gauge the probability values. The extremely low values confirmed the relevancy of the chosen variables.

STEP 3: PREDICTIVE ANALYSIS

Our third and final step involved importing data into a suitable program to construct a **prediction model**. Due to the sheer **volume** of the data considered, normal analysis applications such as MATLAB and Microsoft Excel are not suitable. Hence we use **Apache Spark** along with the **R** programming language (through the sparklyr package) to process the data.

DATA CLEANUP

The initial step in building a predictive model is to **clean u**p the data. This involves removing **undefined null values** and **outliers**.

```
> colSums(is.na(MH))
               X             Date         Latitude        Longitude
               0                0                0                0
 Max.Temperature Min.Temperature     Wind Relative.Humidity
               0                0                0                0
           Solar    Precipitation
               0                0
```

Figure 14: Analyzing null values

We use the built in *colSums(is.na())* function in RStudio to see that there are no null values in the dataset.

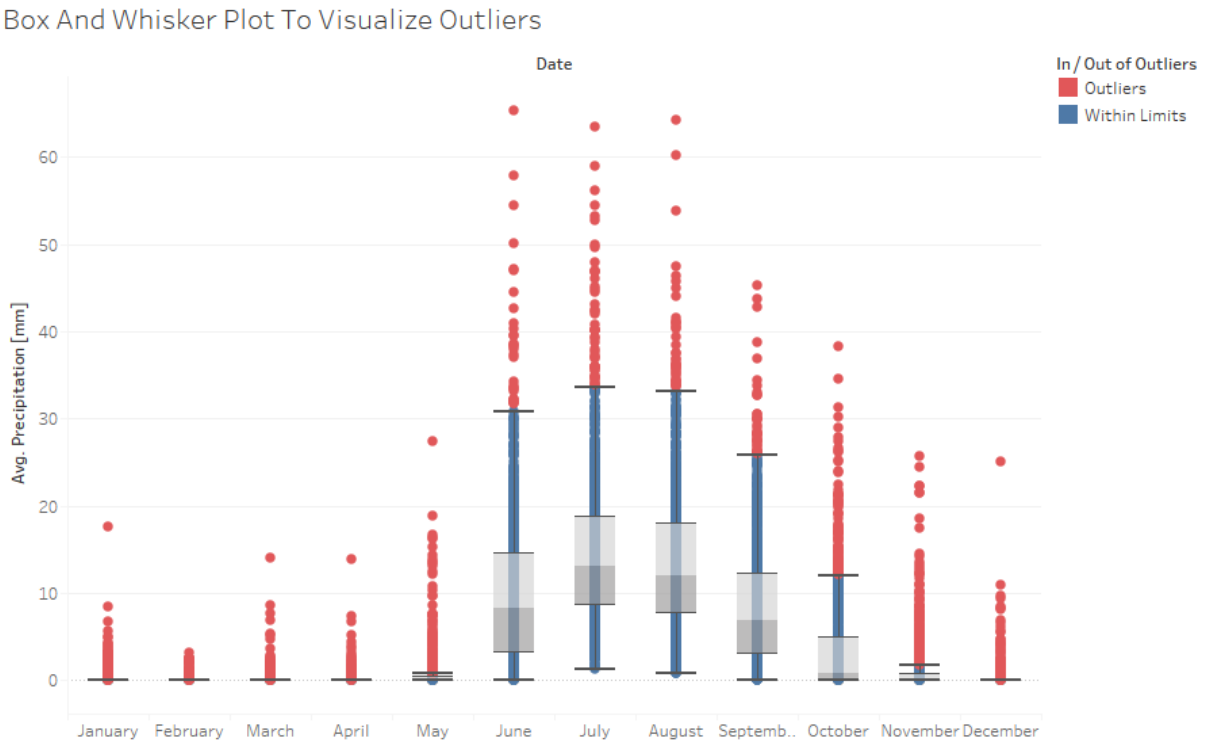

Figure 15: Analyzing outliers

Using tableau, we visualize the **outliers** in a **box and whisker plot** and remove them. In our plot, (Figure 14) each dot represents a single day.

DATA SCALING

After cleaning up null values and outliers, we **scale and center** the data points so as to maintain **zero mean** and **unit variance**. This reduces model computing time and improves model prediction.

```
> summary(MH)
       X                  Date          Latitude      Longitude     Max.Temperature
Min.   :      1   1/1/1980:    309   Min.   :15.77   Min.   :72.50   Min.   :11.21
1st Qu.: 897646   1/1/1981:    309   1st Qu.:18.27   1st Qu.:74.06   1st Qu.:28.30
Median :1795290   1/1/1982:    309   Median :19.51   Median :75.31   Median :31.01
Mean   :1795290   1/1/1983:    309   Mean   :19.26   Mean   :75.82   Mean   :32.16
3rd Qu.:2692935   1/1/1984:    309   3rd Qu.:20.45   3rd Qu.:77.50   3rd Qu.:35.56
Max.   :3590580   1/1/1985:    309   Max.   :21.70   Max.   :80.62   Max.   :55.34
                  (Other) :3588726
Min.Temperature       Wind          Relative.Humidity      Solar        Precipitation
Min.   : 0.80   Min.   : 0.09359   Min.   :0.0304   Min.   : 0.00   Min.   :  0.000
1st Qu.:17.35   1st Qu.: 1.93048   1st Qu.:0.3229   1st Qu.:16.97   1st Qu.:  0.000
Median :21.50   Median : 2.53952   Median :0.5298   Median :20.03   Median :  0.000
Mean   :20.84   Mean   : 2.81174   Mean   :0.5425   Mean   :19.12   Mean   :  4.514
3rd Qu.:24.19   3rd Qu.: 3.40621   3rd Qu.:0.7832   3rd Qu.:23.47   3rd Qu.:  2.438
Max.   :37.63   Max.   :19.14967   Max.   :0.9868   Max.   :29.54   Max.   :421.259
```

Figure 16: Original data values

```
> summary(transformed)
    Latitude          Longitude       Max.Temperature   Min.Temperature       Wind
 Min.   :-2.3293   Min.   :-1.5139   Min.   :-3.8367   Min.   :-4.0779   Min.   :-2.1115
 1st Qu.:-0.6627   1st Qu.:-0.8019   1st Qu.:-0.7052   1st Qu.:-0.7093   1st Qu.:-0.6846
 Median : 0.1706   Median :-0.2323   Median :-0.2098   Median : 0.1346   Median :-0.2115
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.7956   3rd Qu.: 0.7645   3rd Qu.: 0.6241   3rd Qu.: 0.6817   3rd Qu.: 0.4618
 Max.   : 1.6289   Max.   : 2.1886   Max.   : 4.2468   Max.   : 3.4176   Max.   :12.6918
 Relative.Humidity      Solar          Precipitation
 Min.   :-2.01711   Min.   :-3.1038   Min.   :-0.3543
 1st Qu.:-0.86510   1st Qu.:-0.3483   1st Qu.:-0.3543
 Median :-0.04988   Median : 0.1477   Median :-0.3543
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.94800   3rd Qu.: 0.7065   3rd Qu.:-0.1630
 Max.   : 1.74995   Max.   : 1.6923   Max.   :32.7078
>
```

Figure 17: Scaled data values

MODEL GENERATION

After cleaning up and scaling our data, we proceed to construct multiple prediction models in both R and Python using data from 1980 to 2010. After thorough testing, we determine that a random forest model has the best performance.

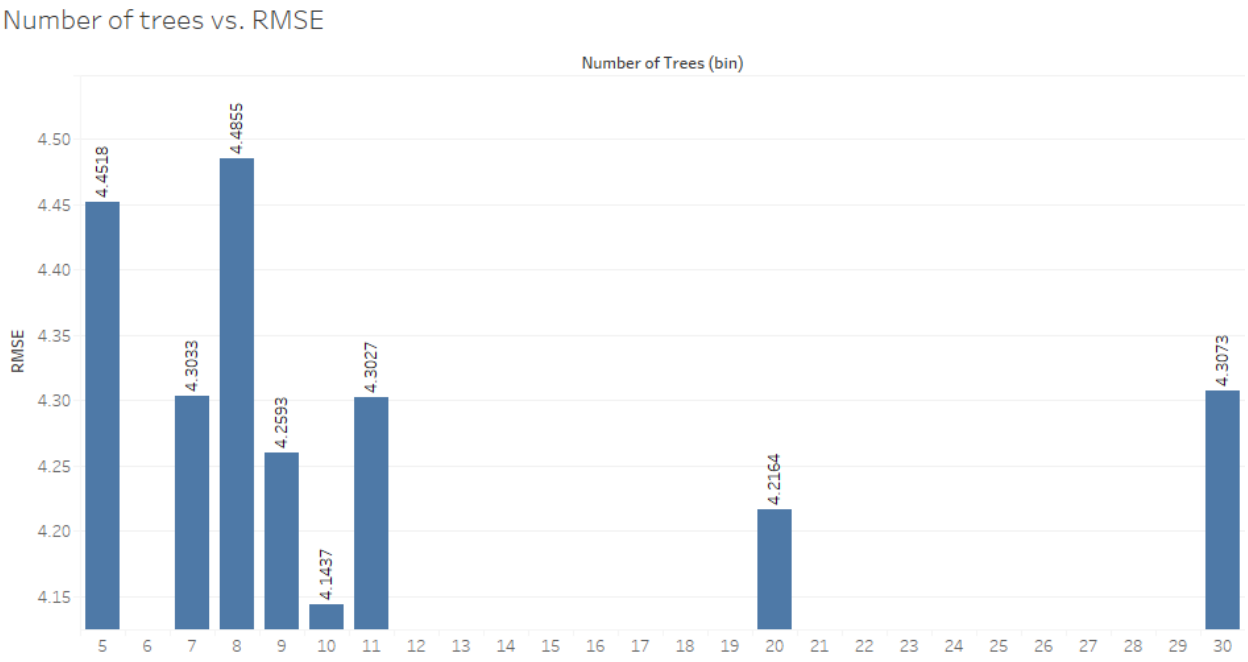

Figure 18: Number of trees vs RMSE

After further testing, we determined that the **Root Mean Square Error (RMSE)** of the model was lowest when using a forest of **10** trees.

MODEL RESULTS

In order to judge the performance of our model, we use testing data from the year 2011, and use this formula to predict the value of precipitation for each day.

We first compare the predicted values to the actual values by using a scatter plot in Figure 17.

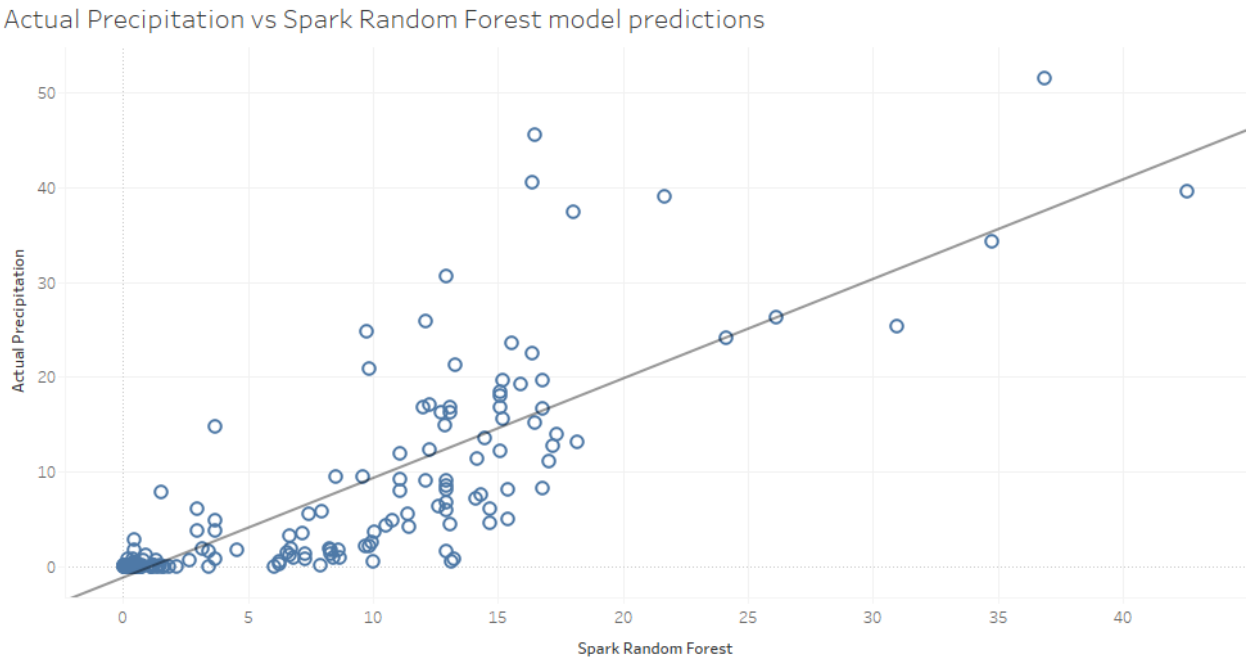

Figure 19: Actual vs Predicted values

The closer the trend line is to 45˚, the better the prediction.

We also compare the monthly average predictions in Figure 18.

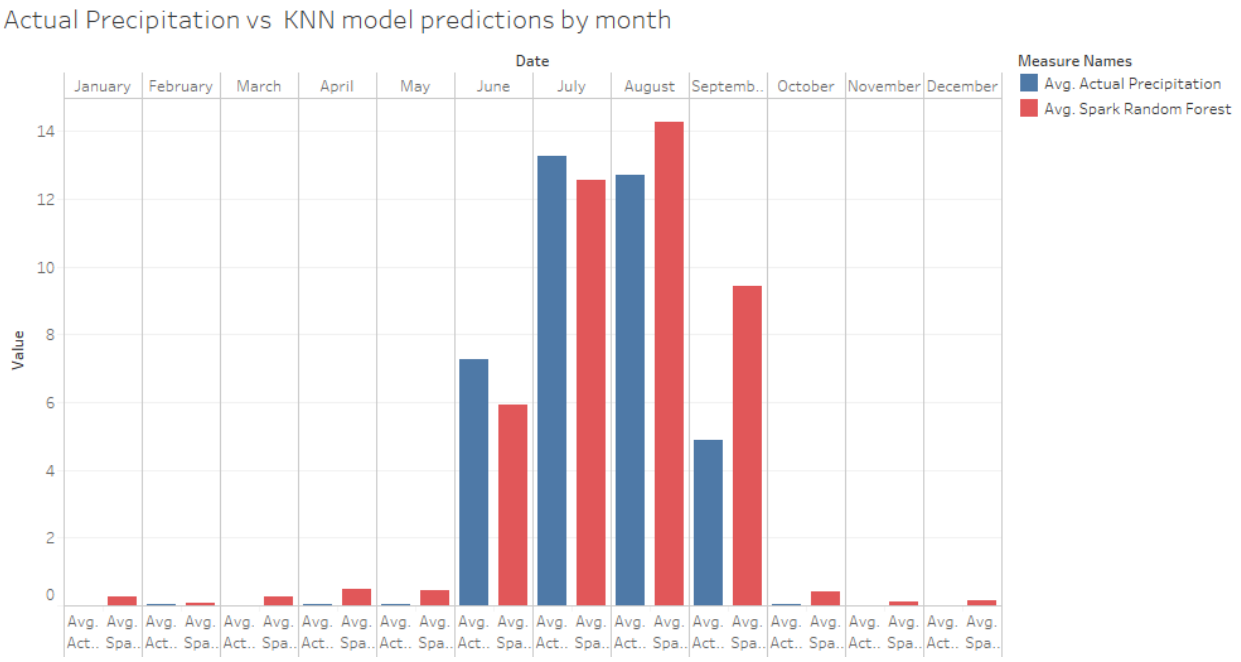

Figure 20: Monthly average predictions

Our model is able to predict rainfall with an average error of ±1.9 mm.

GUI IMPLEMENTATION

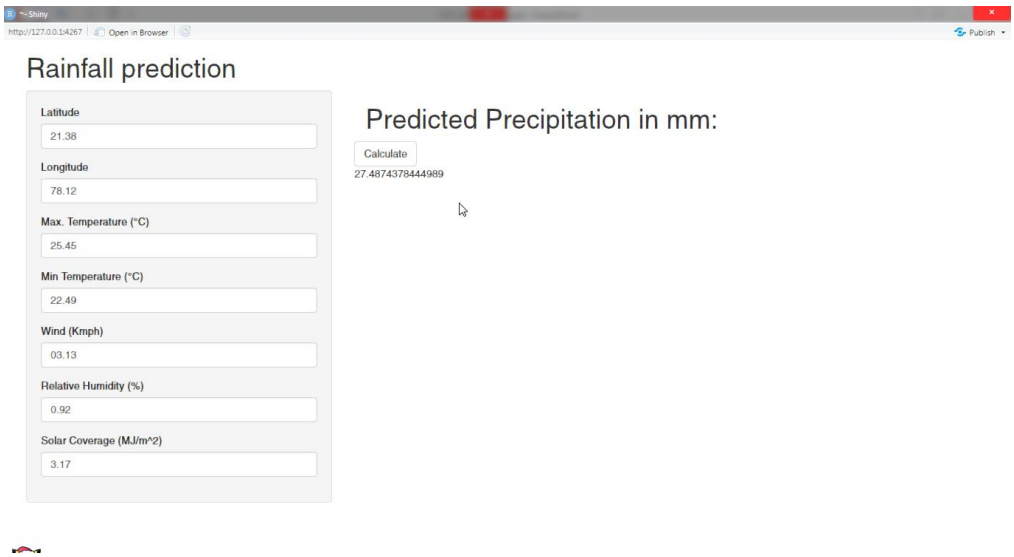

Figure 21: GUI

We utilized the shiny package for R to build a basic **GUI** which allows a user to input the independent values and predicts the output with the click of a button.

FUTURE SCOPE

We aim to improve our prediction by utilizing **better modelling techniques** and market this program in various countries.

We also plan on expanding this project to make it more versatile in terms of **automatic data acquisition** and **user notification**.

CONCLUSION

Through this research project we studied the importance of rainfall, its measurement techniques, and their shortcomings and devised a solution to overcome these by integrating **predictive analytics** techniques from the new and upcoming field of **big data** analytics.

ACKNOWLEDGEMENT

This project would not have been possible without the guidance of our professors who were always ready to answer our doubts and suggest novel solutions to our problems.

BIBLIOGRAPHY

Dile, Y. T., R. Srinivasan, 2014. Evaluation of CFSR climate data for hydrologic prediction in data-scarce watersheds: an application in the Blue Nile River Basin. *Journal of the American Water Resources Association (JAWRA)* 1-16. DOI: 10.1111/jawr.12182

Fuka, D.R., C.A. MacAllister, A.T. Degaetano, and Z.M. Easton. 2013. Using the Climate Forecast System Reanalysis dataset to improve weather input data for watershed models. *Hydrol. Proc.* DOI: 10.1002/hyp.10073.

Lily Ingsrisawang ET. Al. Machine Learning Techniques for Short-Term Rain Forecasting System in the Northeastern Part of Thailand

National Centers for Environmental Prediction (NCEP) Climate Forecast System Reanalysis (CFSR) globalweather.tamu.edu

APPENDIX

FIGURES

TABLES

CODE

R (MODEL CONSTRUCTION)

```
library(sparklyr) #Import sparklyr package to integrte Spark and R

sc <-

  spark_connect(master = 'local') #Connect to a local spark instance
```

```
test_tbl <-

  spark_read_csv(sc, "test", "C:/Users/user/Documents/VIT/Sem 6/EDI/test.csv") #Read the testing dataset directly into a Spark DataFrame

train_tbl <-

  spark_read_csv(sc,

                 "train",

                 "C:/Users/user/Documents/VIT/Sem 6/EDI/train.csv") #Read the testing dataset directly into a Spark DataFrame

actual <-

  read.csv("C:/Users/user/Documents/VIT/Sem 6/EDI/res.csv") #Read the validation dataset



#Model Generation, random forest using 10 trees

fit <- train_tbl %>%

  ml_random_forest_regressor(

    response = "Precipitation",

    features = c(

      "Latitude",

      "Longitude",

      "Max_Temperature",

      "Min_Temperature",
```

```
    "Wind",
    "Relative_Humidity",
    "Solar"
  ),
  num_trees = 10
)

pred <-
  ml_predict(fit, test_tbl) %>% collect  #Generation predioctions on testing set

eval_tbl <-
  as.data.frame(cbind(pred$prediction, actual$Actual.Precipitation)) #Create evaluation
DataFrame using validation dataset and predictions

eval <-
  copy_to(sc, eval_tbl, overwrite = TRUE) #Copy validation DataFrame to Spark DataFrame

#Evaluate performance of model
eval <- ml_regression_evaluator(
  eval,
  label_col = 'V2',
```

```
  prediction_col = 'V1',

  metric_name = 'rmse',

  uid = 'linear regression'

)

print(eval)

ml_save(fit, "C:/Users/user/Documents/VIT/Sem 6/EDI/MODEL") #Export model for ease of use in GUI
```

PYTHON (MODEL CONSTRUCTION)

```
import numpy as np

import pandas as pd

import scipy

import matplotlib.pyplot as plt

from pylab import rcParams

import urllib

import sklearn

from sklearn.neighbors import KNeighborsRegressor

from sklearn import neighbors

from sklearn import preprocessing

from sklearn.model_selection import train_test_split

from sklearn import metrics

from sklearn import utils

df=pd.read_csv(r"C:\Users\Aneesh\Desktop\MH.csv")

df_prime=df.ix[:,(4,5,6,7,8)].values
```

```
y_prime=df.ix[:,9].values

clf=neighbors.KNeighborsRegressor(5)

clf.fit(df_prime,y_prime)

print(clf)

pred=pd.read_csv(r"C:\Users\Aneesh\Desktop\Testing.csv")

pred_prime=pred.ix[:,(3,4,5,6,7)].values

y_test=pred.ix[:,8].values

y_pred=clf.predict(pred_prime)

print(y_pred)

pred.to_csv('knn.csv',columns=['Actual Precipitation'])

i=0

for i in range(i,365):

   print(y_pred[i])
```

R (GUI CONSTRUCTION)

```
library(shiny) #Import shiny library to create GUI

#Define rontend UI design

ui <- fluidPage(

 headerPanel("Rainfall prediction"),

 sidebarPanel(

   numericInput("Latitude", "Latitude", value = 0),

   numericInput("Longitude", "Longitude", value = 0),

   numericInput("Max_Temperature", "Max. Temperature (°C)", value = 0),

   numericInput("Min_Temperature", "Min Temperature (°C)", value = 0),

   numericInput("Wind", "Wind (Kmph)", value = 0),
```

```
  numericInput("Relative_Humidity", "Relative Humidity (%)", value = 0),
  numericInput("Solar", "Solar Coverage (MJ/m^2)", value = 0)
 ),
 mainPanel(
  headerPanel("Predicted Precipitation in mm: "),
  actionButton("calc", "Calculate"),
  textOutput("optext")
 )
)
#Define backend server functions
server <- function(input, output)
{
 observeEvent(input$calc,
          {
            print("Initializing library")
            library(sparklyr)
            print("Creating table")
            temp_tbl <- data.frame(
              "Latitude" = input$Latitude,
              "Longitude" = input$Longitude,
              "Max_Temperature" = input$Max_Temperature,
              "Min_Temperature" = input$Min_Temperature,
              "Wind" = input$Wind,
              "Relative_Humidity" = input$Relative_Humidity,
```

```
              "Solar" = input$Solar
            )
            print("Connecting to local spark cluster")
            sc <- spark_connect(master = 'local')
            print("Creating spark dataframe")
            userinput <-
              copy_to(sc, temp_tbl, overwrite = TRUE)
            print("Loading model")
            model <-
              ml_load(sc, "C:/Users/user/Documents/VIT/Sem 6/EDI/MODEL")
            print("Predicting on user variables")
            pred <- ml_predict(model, userinput) %>% collect
            print("Done")
            output$optext <- reactive(max(0, pred$prediction))
          })

}
shinyApp(ui, server) #Run app using UI and Server
```