# Time Series Analysis on Weather Data

Harpal Kaur Dhindsa[1], Niveda Mudaliar[2], Palak Sharma[3], Carlton Daniel[4], Mr. Pravat Kumar Jena[5]

[1,2,3,4] .M.Tech CSE with Big Data specialization ,SCOPE, VIT University

[5].Assistant Professor, Sr. Grade 1, SCOPE, VIT University

## Abstract

Climate change is affecting today's world. Various climatic parameters are the factors affecting the climate. Changing in rainfall patterns, hot atmosphere, unusual seasonal change are the impacts of this climate change. In an agricultural country like India, rainfall is very important. Rainfall is a critical parameter affecting different situations in many countries. To predict certain natural calamities like floods, the prediction of rainfall is evident. India is economically driven by agriculture, which is why rainfall patterns play an important role and hence, more precise rainfall prediction techniques in India are necessary. Hence, the prediction analysis of rainfall is very much needed. Data mining and analysis of meteorological data helps to find hidden trends, patterns, etc. In this paper, we have collected the weather dataset of different Indian cities of the last 10 years to perform exploratory data analysis and clustering using K-means and PCA (Principal Component Analysis). We propose to perform rainfall prediction analysis using different machine learning algorithms like Linear Regression, Decision tree regression and Gradient Boosted regression. The weather data is then classifying into different levels like no rain, moderate rain, heavy or very heavy rain using different classification problems like Support Vector Machine (SVM), Logistic Regression, Decision tree and Random Forest.

## Introduction

Nowadays, climatic change has adverse effects on the globe. The atmosphere and the earth are getting warmer. The changing in rainfall patterns, irregular change in seasons impact the climate change a lot. Irregular rainfall affects the agriculture sector and the farmers. Rainfall prediction helps farmers to manage their crops and take necessary actions in advance. It also helps in managing the country's economy. Forecasting the rainfall with the help of visual representations helps us to know about the trends or the rainfall patterns.

The main objective of this paper is to perform exploratory data analysis on weather data to understand more about the dataset and get useful insights from them. Clustering and dimensional reduction is carried out on the dataset by using K-means and Principal Component Analysis (PCA) respectively. The clusters are formed city wise on the basis of different parameters. In this paper, we propose to perform rainfall prediction analysis on this weather data using machine learning algorithms like Linear regression, Decision Tree regression and Gradient Boosted regression. Then, we are binning the data and classifying into different levels of rainfall like no rain, moderate and heavy rain for every Indian city.

Many researchers have been done regarding time series analysis and rainfall prediction. In the paper [1], the authors have proposed a time series-based rainfall prediction analysis using linear regression model to predict rainfall in Kerala. The authors Ashwini and Kalaivani in their paper [2], have used ARIMA model to predict seasonal rainfall. However, the accuracy obtained was better than the

regression model. In the paper [3], Grace and Suganya have proposed to use multiple linear regression model to predict rainfall with the help of correlation, accuracy and other parameters.

## Data Description

The dataset we have used for performing this time series analysis is extracted from World Weather online through API. The dataset extracted contains weather data of different Indian cities for the past 10 years (2010-2020). There are around more than 717552 records. It includes around 25 different climatic parameters like date-time, max_temp, min_temp, humidity, precipitation, pressure, location, wind_pressure, etc.

## Implementation

To understand the flow of data and discover the patterns in the data, first the correlation between the features presents in the data need to be found. The correlation plot for all the features present in the data is displayed in the figure below.

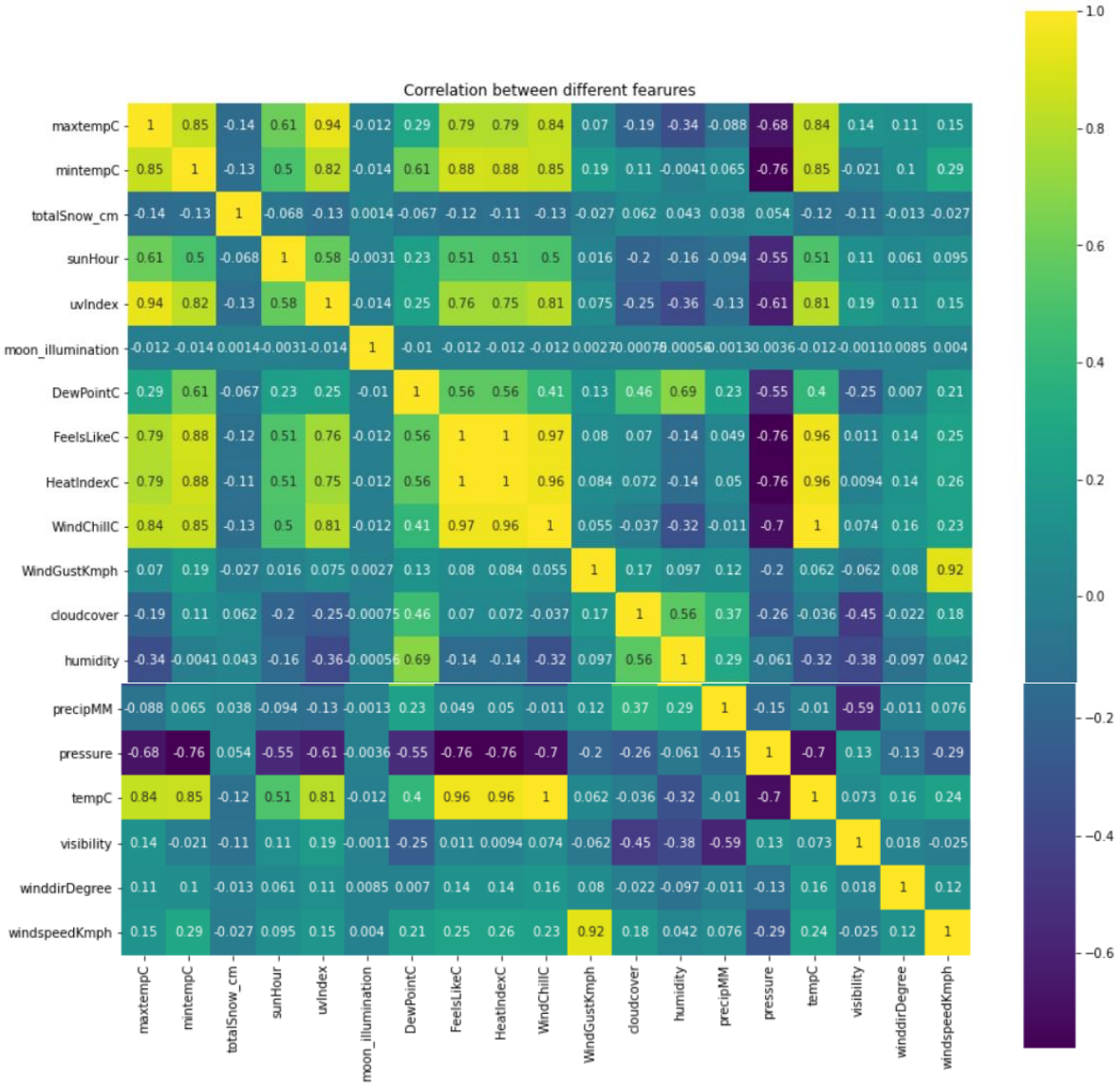


Figure 1: Correlation plot for all features

From the correlation plot we find that the following columns are highly correlated, maximum temperature, minimum temperature, UV index, Feels like temperature in Celsius, heat index, wind chill temperature in Celsius, actual temperature, sun hour, humidity, precipitation, and dew point.

If we extract a few of these important features and show a correlation plot, the illustration is exhibited in figure below.

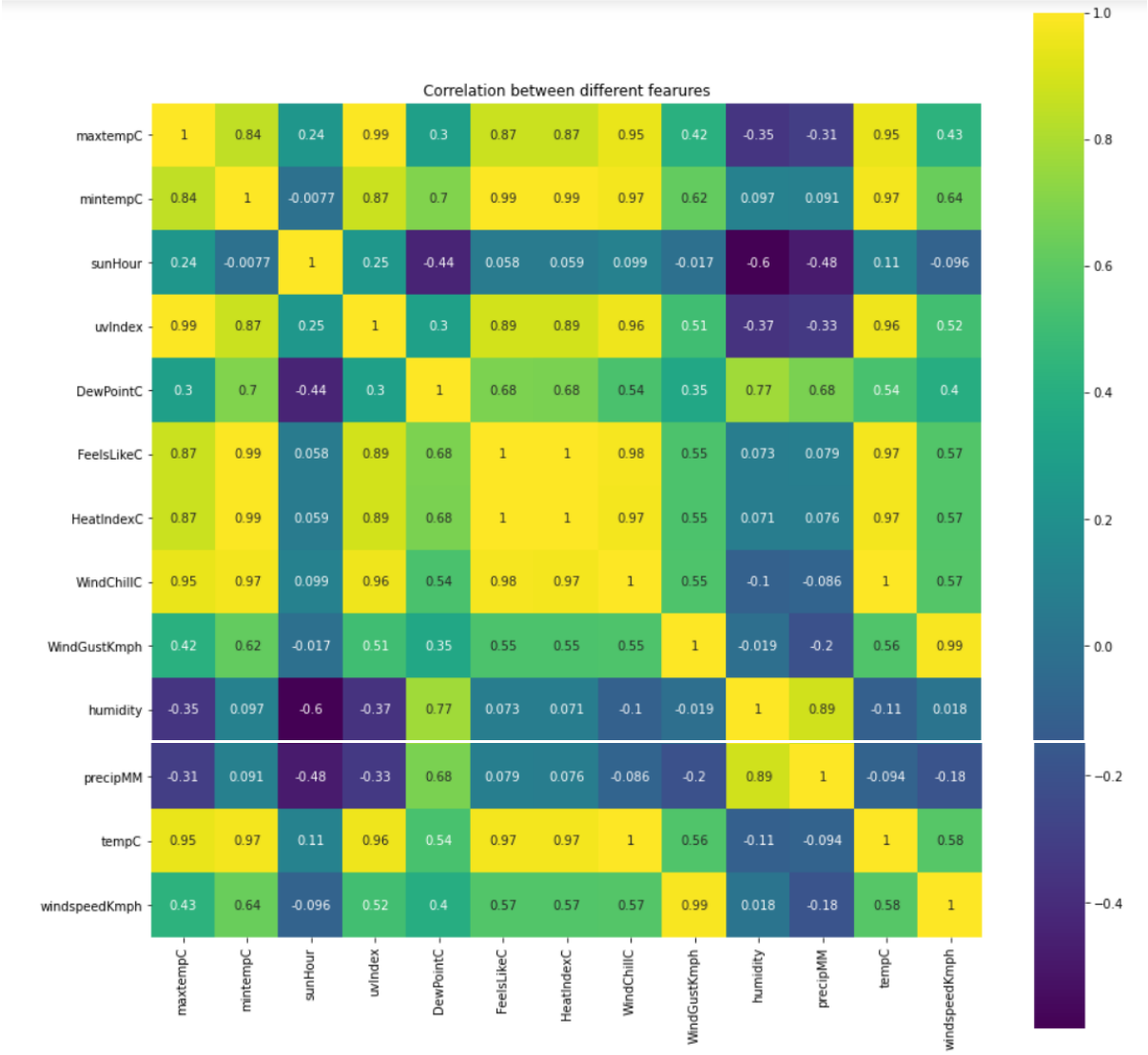



Figure 2: Correlation plot for selective features

The data consists of Date and time column, in which the data is collected over a period of 10 years and, it is taken an interval of 3 hours. To be able to understand the distribution of data over a period of time, we plot a hourly scatter plot on the precipitation column, as shown in the figure below.

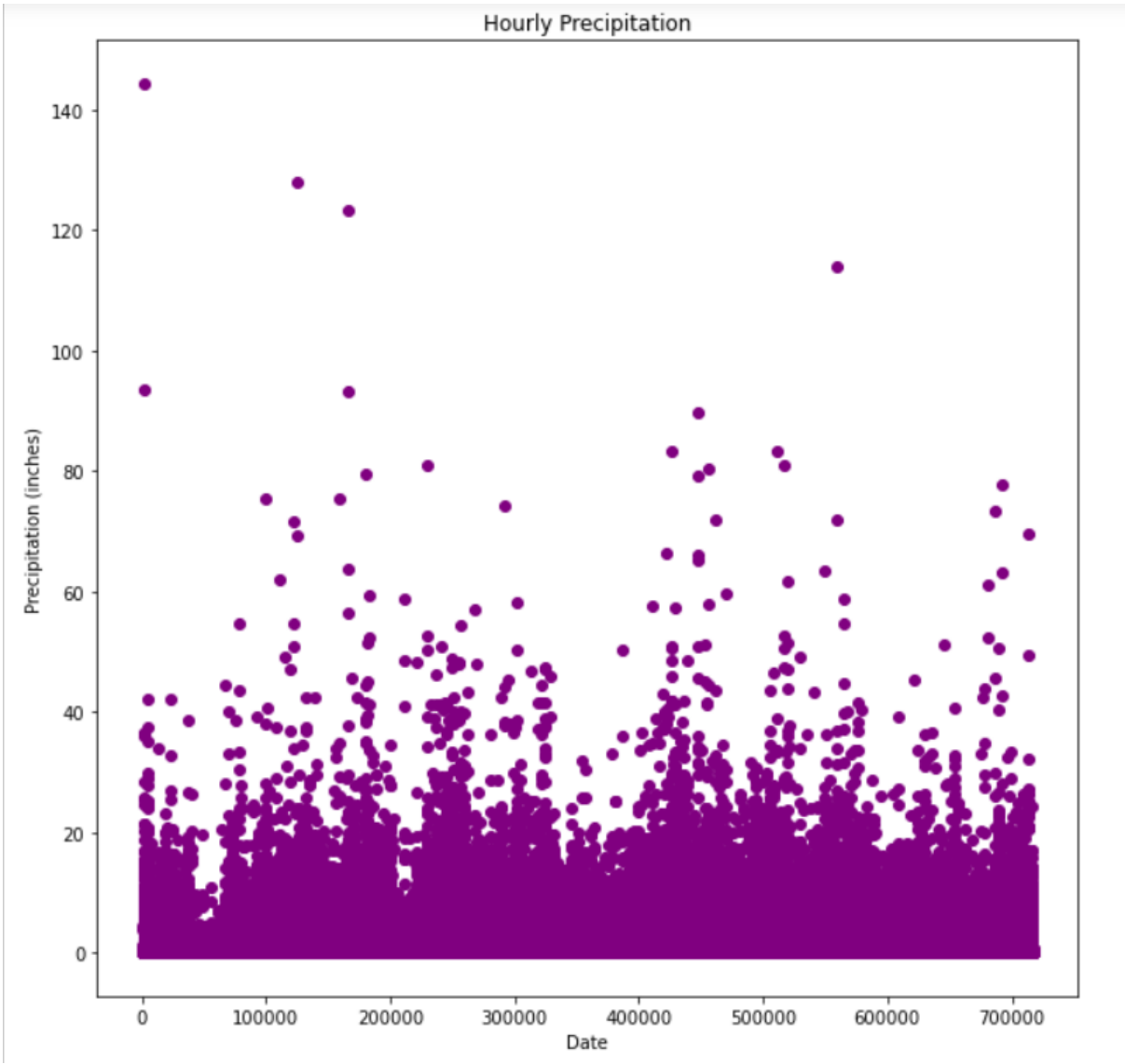



Figure 3: Hourly Precipitation

If we want to find the mean precipitation for a particular location, it can be demonstrated in the figure below.

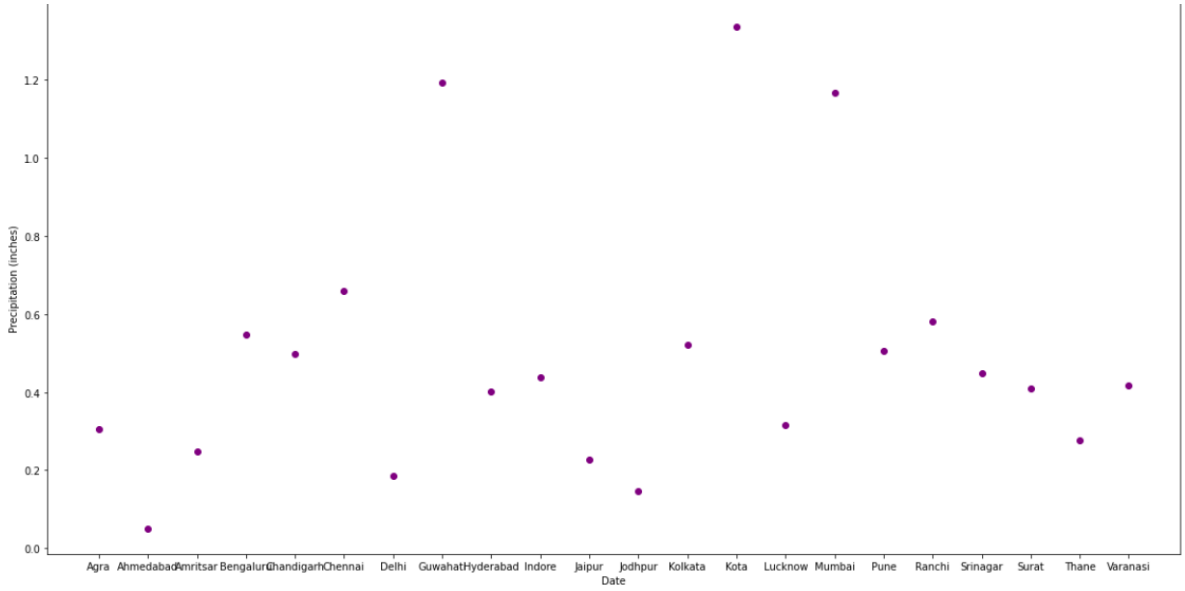
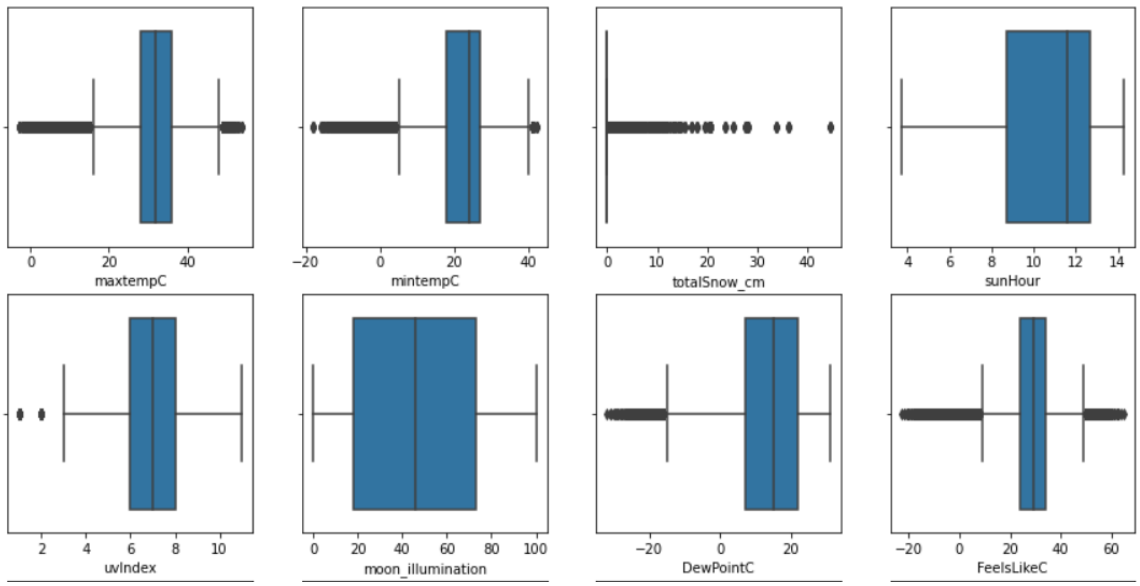



Figure 4: Mean precipitation for every Location

For data cleaning and preprocessing, we need to check if the data consists of outlier. To demonstrate that the box plots are shown in the figure below.



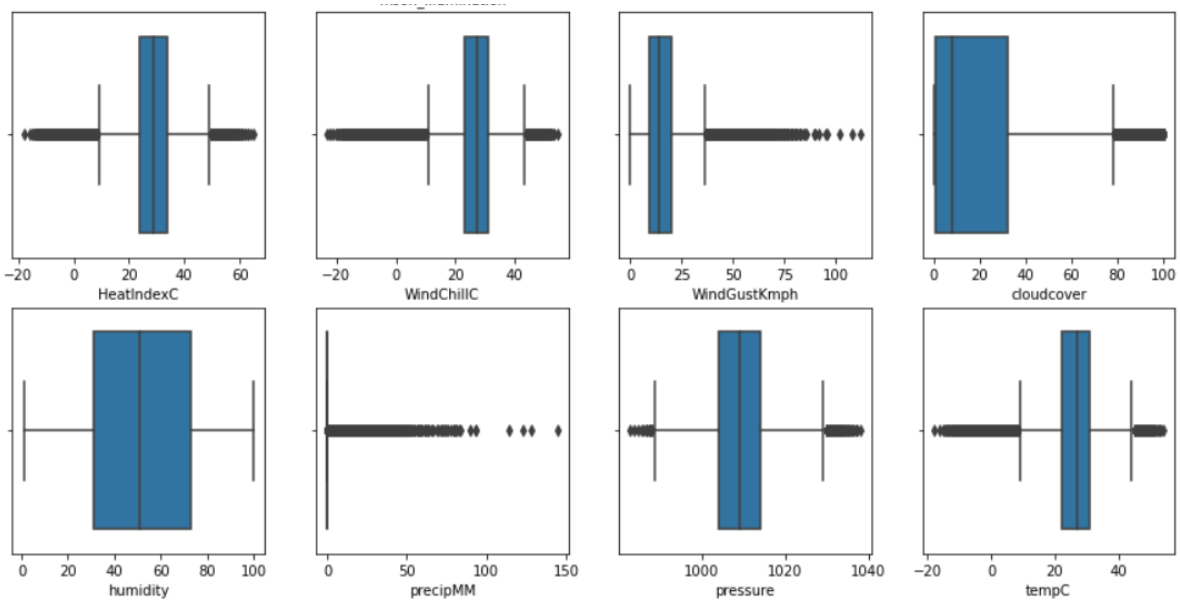
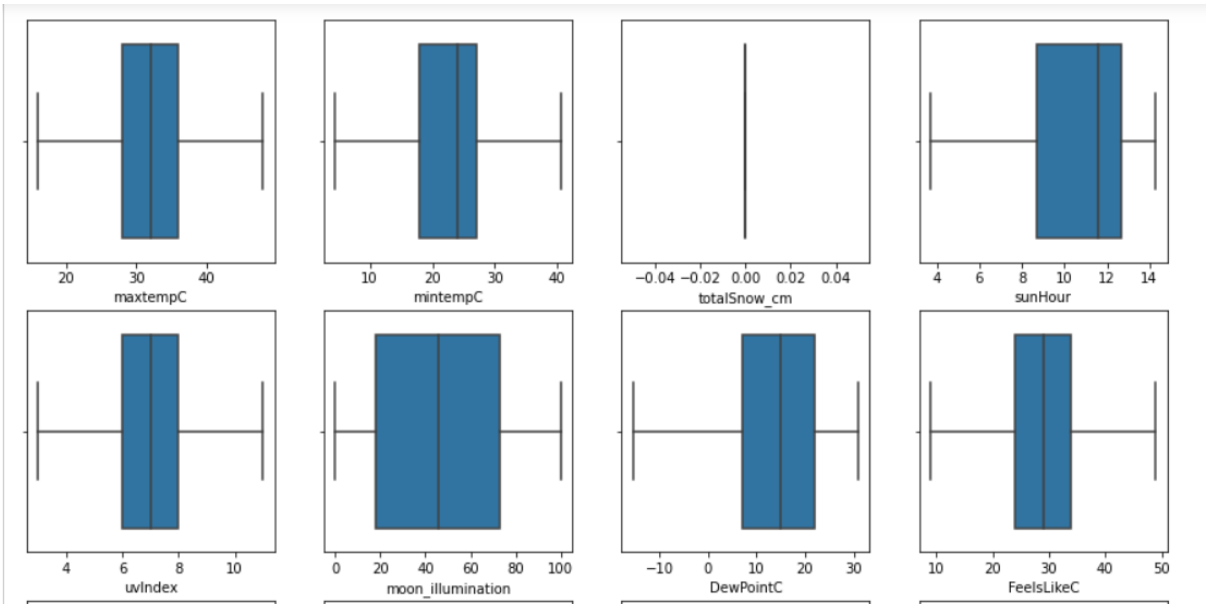



Figure 5: Box Plot outlier detection

After handling the outliers, the resultant box plots are shown in the figure below.



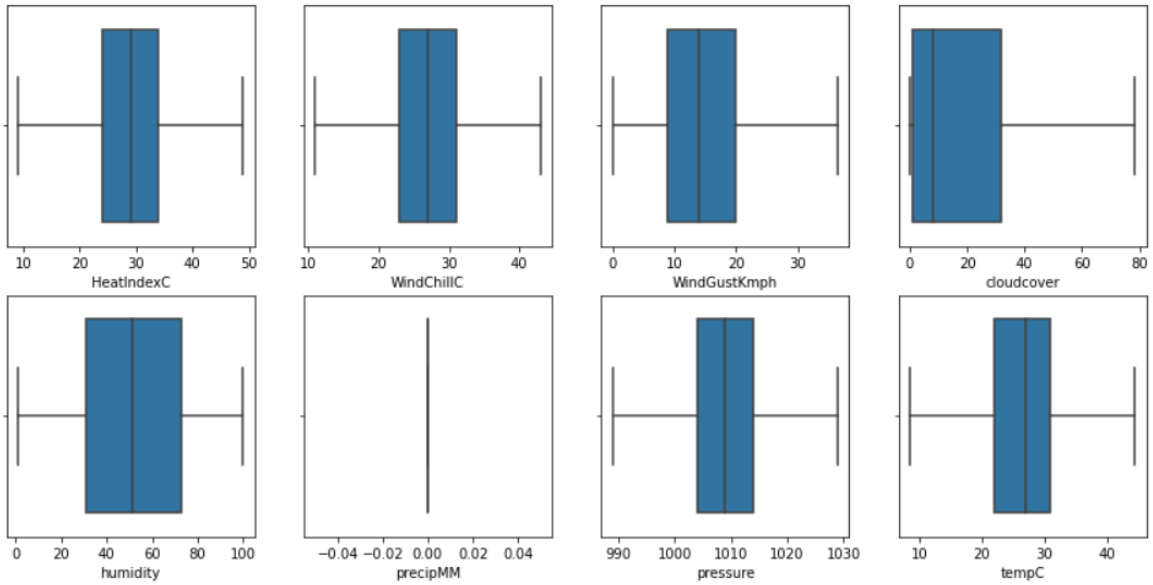



Figure 6: Box plot after removing outliers

**K-Means:**

To be able to cluster the locations on the basis of similarity of traits between them, we use K-means algorithm to cluster them.

To process the data, the K-means algorithm starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative calculations to optimize the positions of the centroids

It halts creating and optimizing clusters when either:

- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

In order to perform K-means we select the temperature, sun hour, and humidity for all the locations.

The below figure shows the Elbow plot to understand how many clusters would be best to be able to cluster the data.

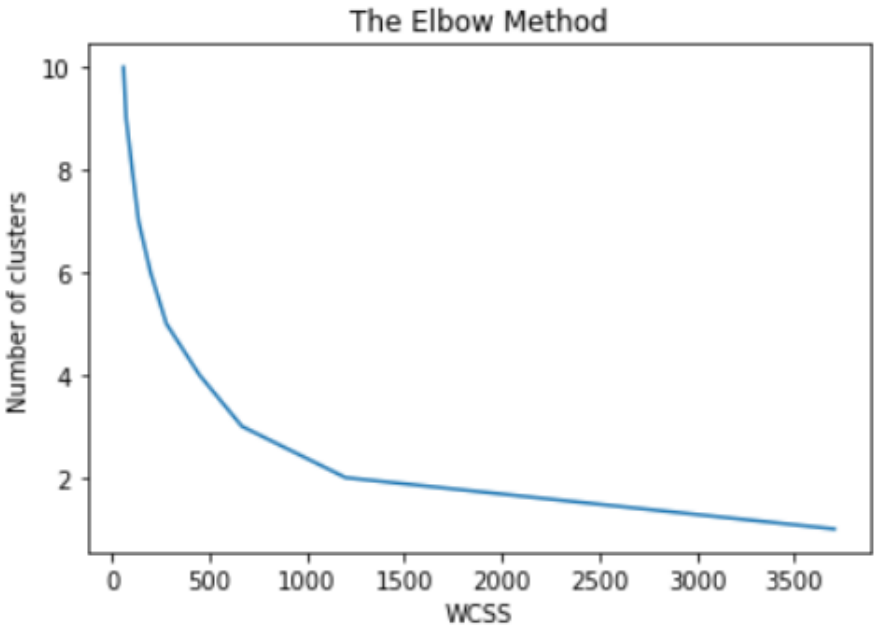



Figure 7: The Elbow Method

The below figure shows the k-means on the basis of humidity and temperature. The red plus signs are the centroids for the clusters.

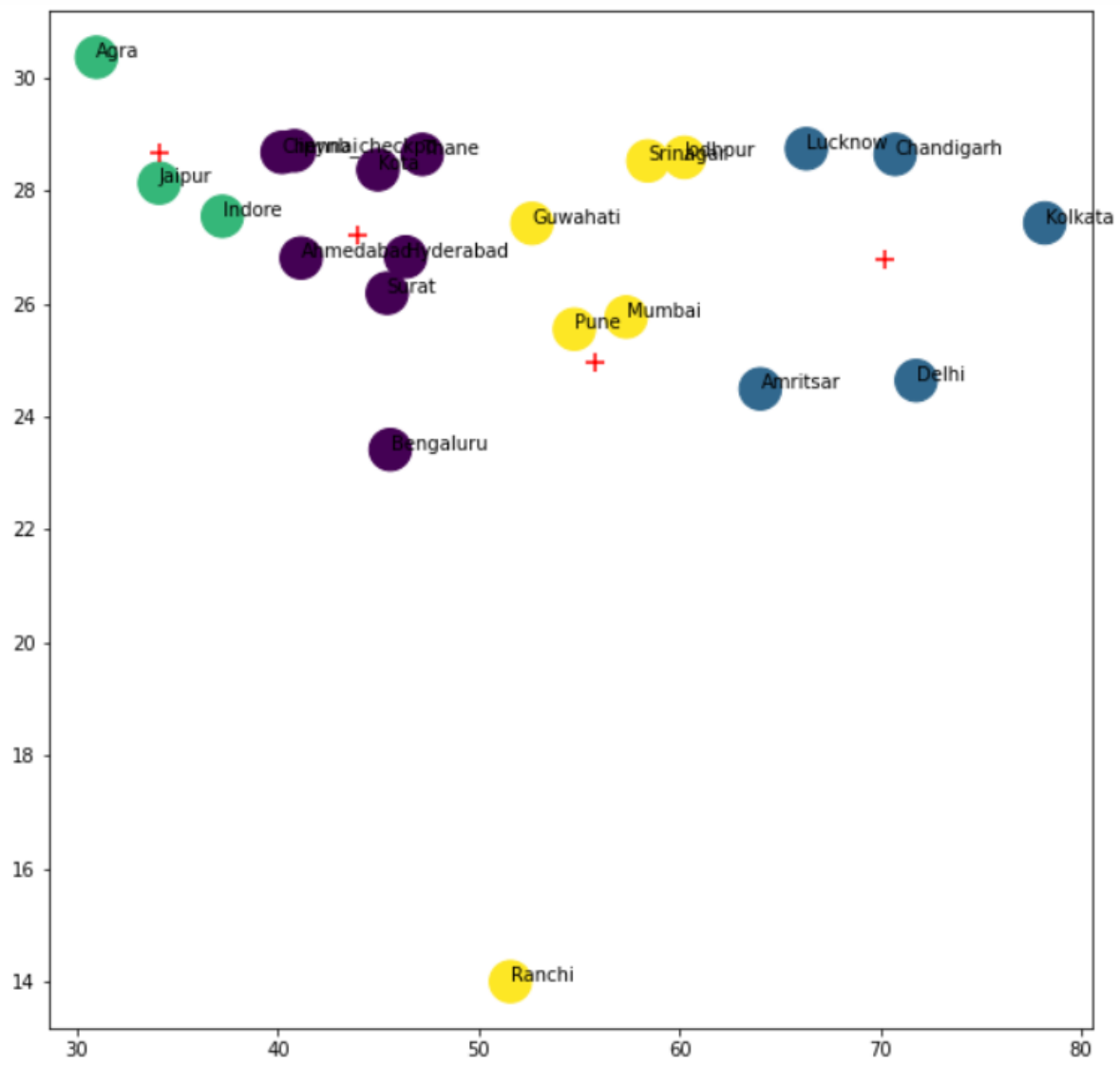



Figure 8: K-means: Humidity Vs Temperature

The below figure shows the k-means on the basis of sun hour and temperature. The red plus signs are the centroids for the clusters.

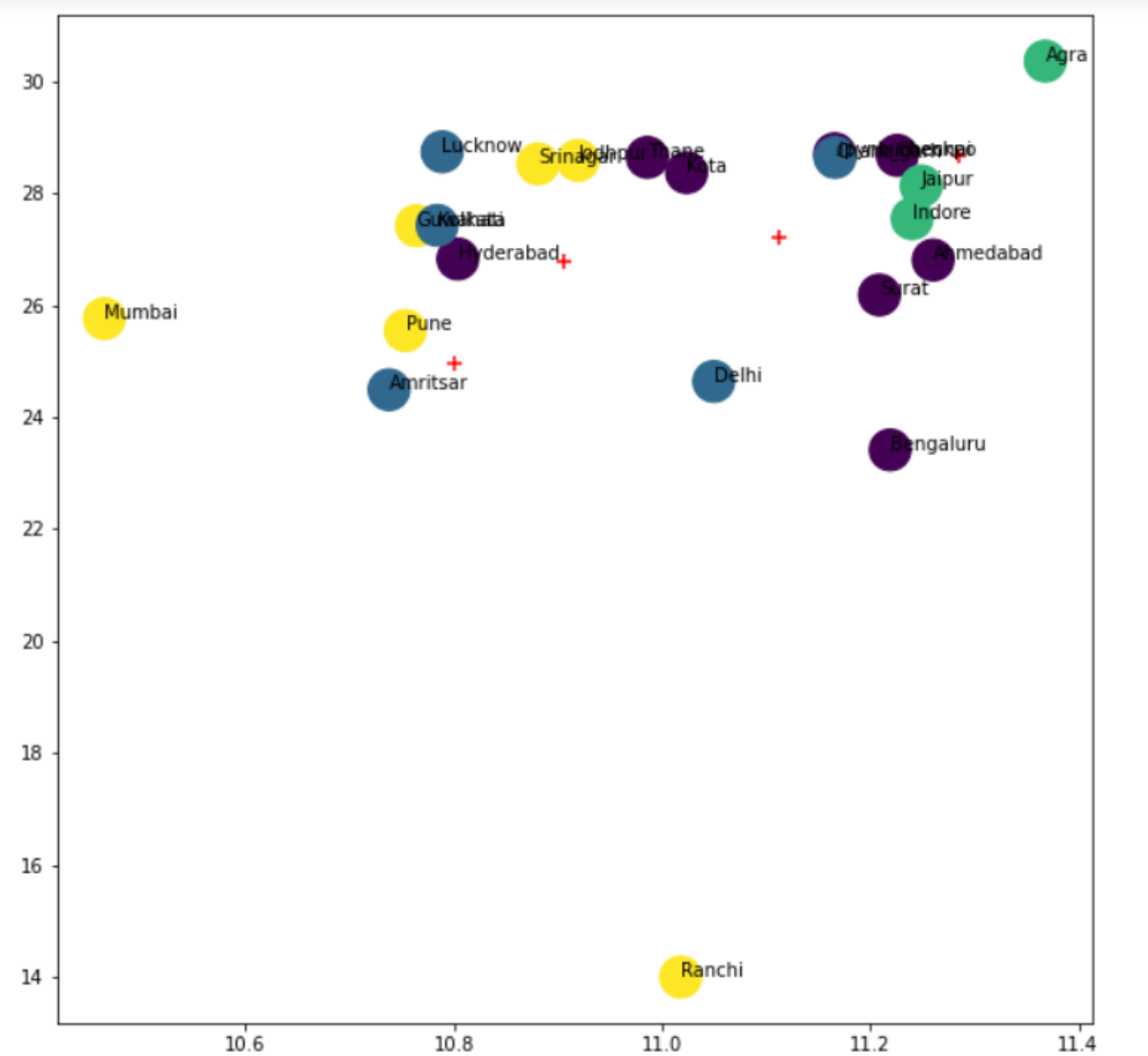



Figure 9: K-means: sun hour Vs Temperature

The below figure shows the k-means on the basis of sun hour and humidity. The red plus signs are the centroids for the clusters.

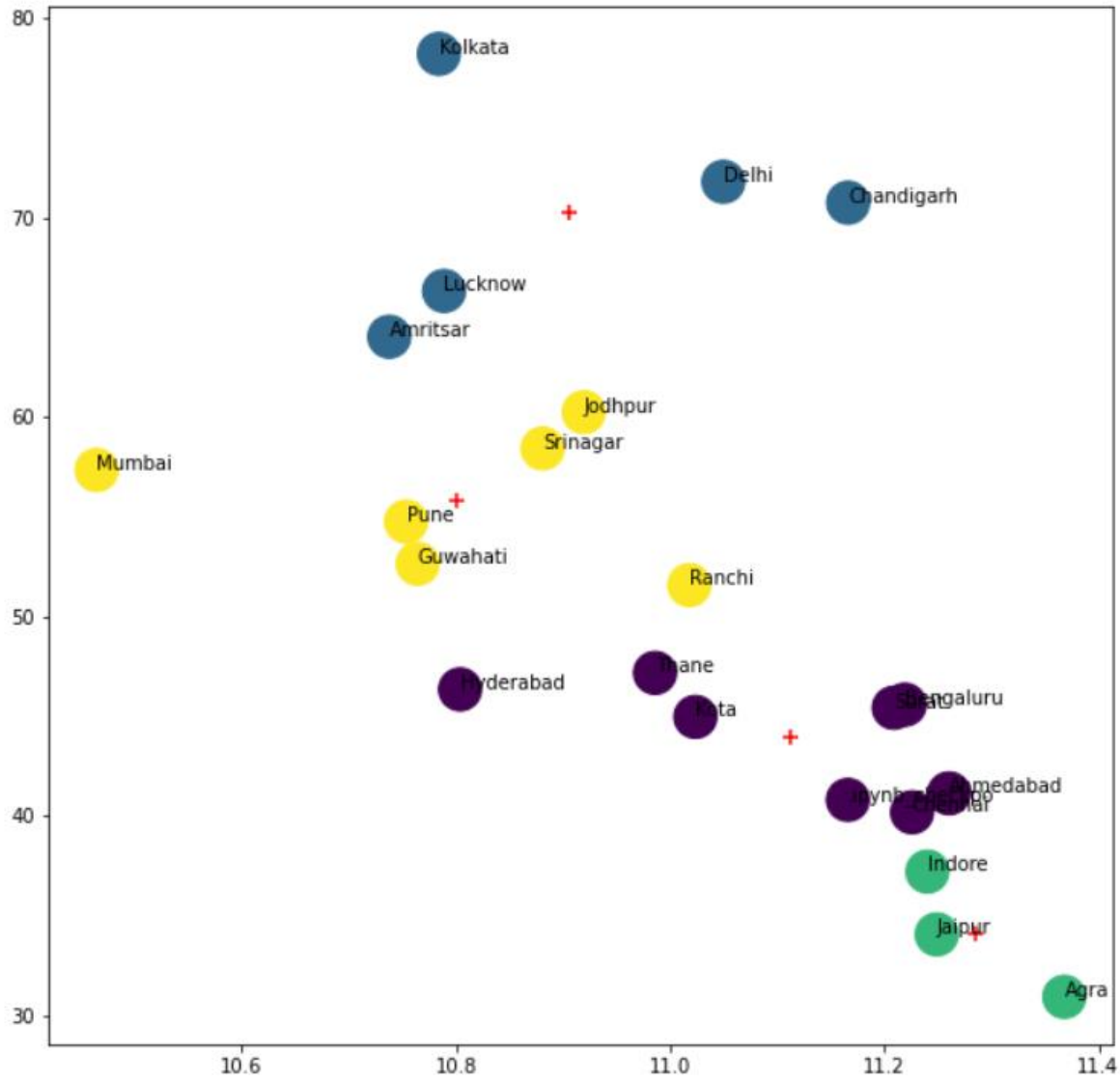



Figure 10: K-means: Sun hour Vs Humidity

As data consists of date time in hourly basis, for further processing we will group all the date time data to convert to a daily basis.

The below plots demonstrate the variation in groups in individual categorical columns, sunrise, and sunset.

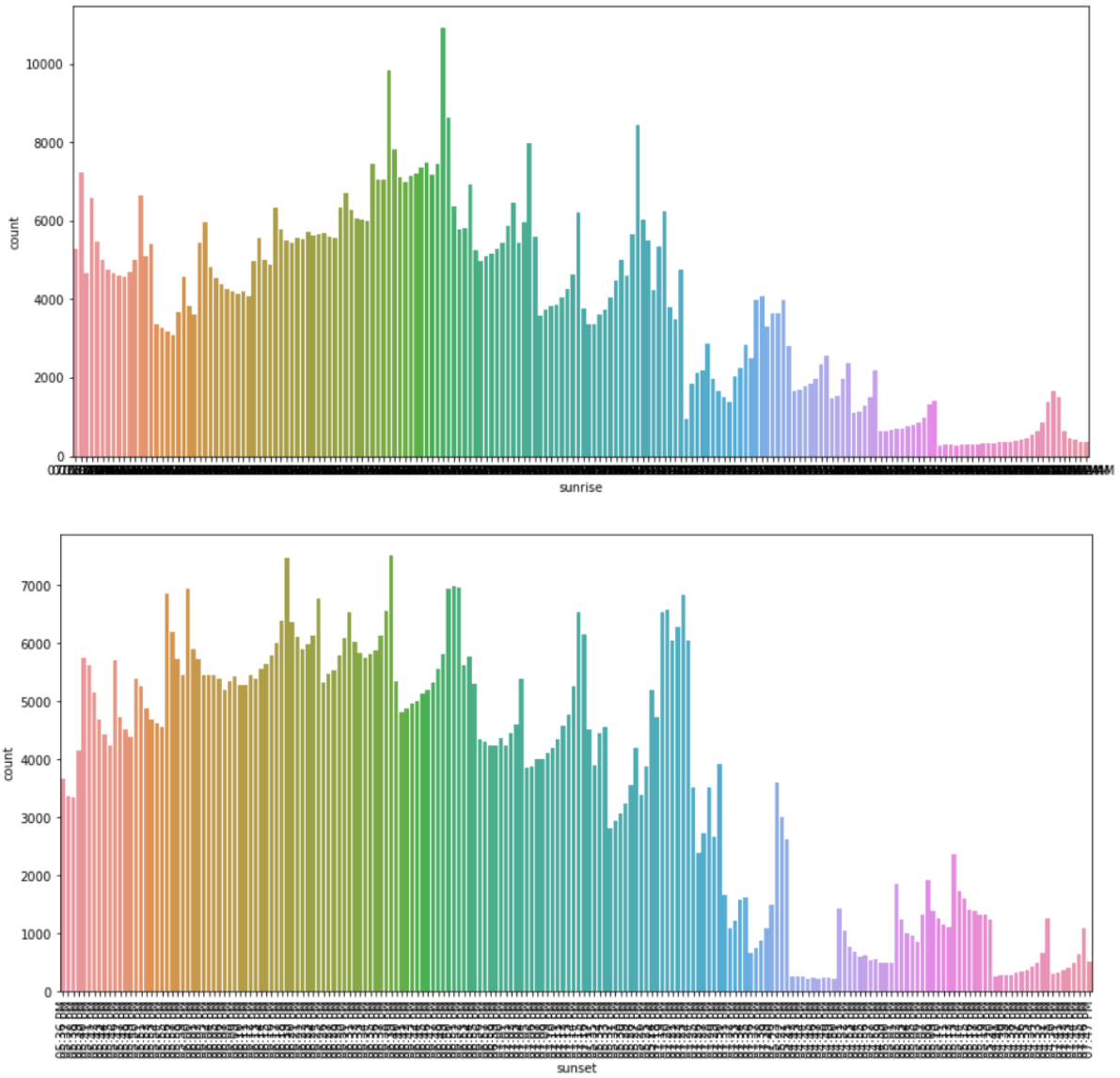



Figure 11: Check variation in groups in individual categorical columns

**Principal Component Analysis (PCA):**

Since the data consists of 19 features in the dataset, to understand which feature will be more useful as compared to other features we apply PCA.

PCA is used to reduce dimensionality to obtain an economic description of the weather profile, with differences between various profiles. The use of PCA helps to recognize patterns by explaining the variance of a large set of intercorrelated variables, and it transforms them into a smaller set of independent factors. Also, it reduces the computational complexity of the model which makes machine learning algorithms run faster.

Steps that are involved in PCA are:

1. Standardizing the data. (with mean =0 and variance = 1)
2. Computing the Covariance matrix for all features.
3. Attain the Eigenvectors and Eigenvalues from the covariance matrix.

4. Sort the obtained eigenvalues in descending order and select the top k Eigenvectors that correspond to the k largest eigenvalues (k will become the number of features of the new dimension subspace k≤d, d is the number of original features).
5. Construct the projection matrix W from the carefully chosen k Eigenvectors.
6. Transform the original data set X by means of W to obtain the new k-dimensional feature subspace Y.

As the first step of PCA is standardization, we will first standardize the numerical data and encode the categorical data such as location, sunrise, sunset, etc. The below plot shows the sunrise and sunset data after encoding.

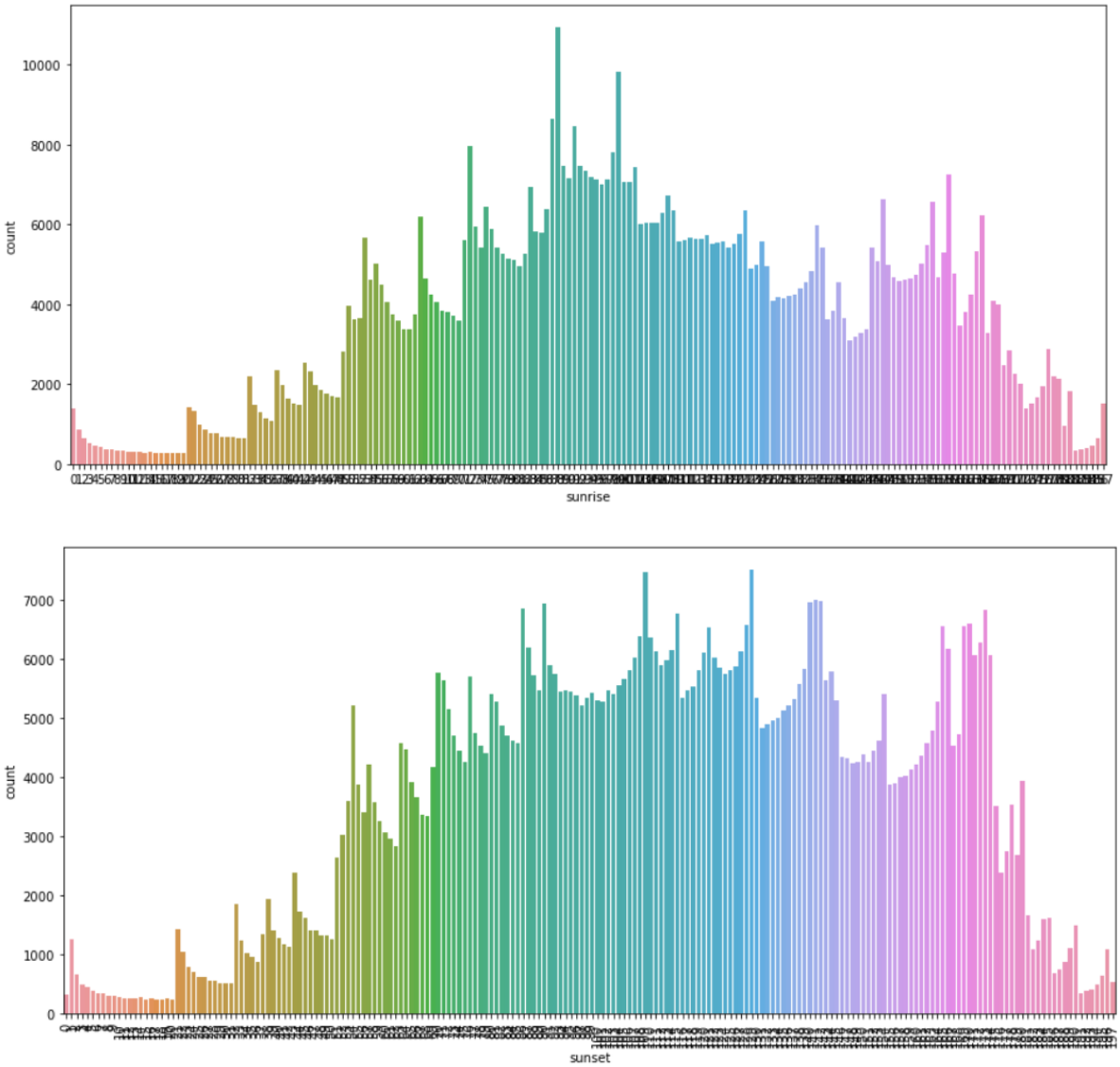



Figure 12: Sunrise and Sunset variations after encoding

The below figure is an elbow plot for the PCA, it shows the number of components from which the variance becomes linear, that is it shows the peak from which number of components can be selected from. When the variance reaches its peak, it shows the number of components due to which maximum variance is reached.

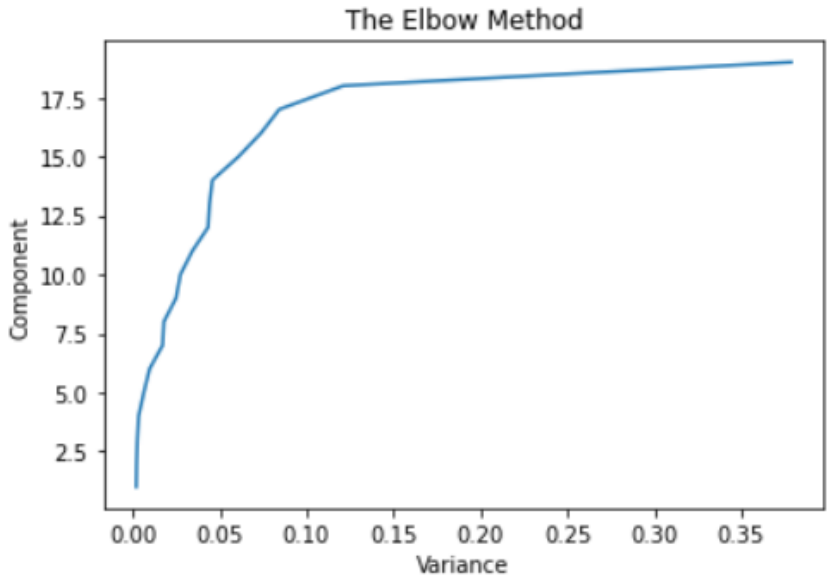
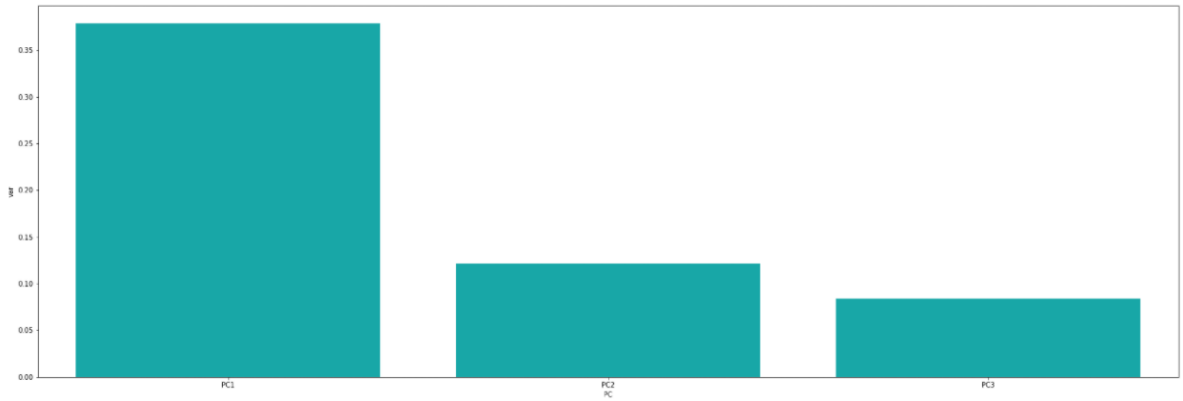



Figure 13: PCA Elbow plot

The below figure shows the PCA generated when number of components is taken as 3. All the features are taken into account. The plot shows that the variance of the first principal component is more weighted in understanding the data.




| | var | PC |
|---|---|---|
| **0** | 0.378587 | PC1 |
| **1** | 0.120937 | PC2 |
| **2** | 0.084200 | PC3 |

Figure 14: Individual explained variance

The below figure shows the PCA generated when number of components is taken as 3. The features taken into account are temperature, humidity, and precipitation. As the plot shows the PC 1 shows 92.4% variance hence it is most important, PC 2 shows 7.6% it is comparatively less important, but PC 3 is useful to the least.

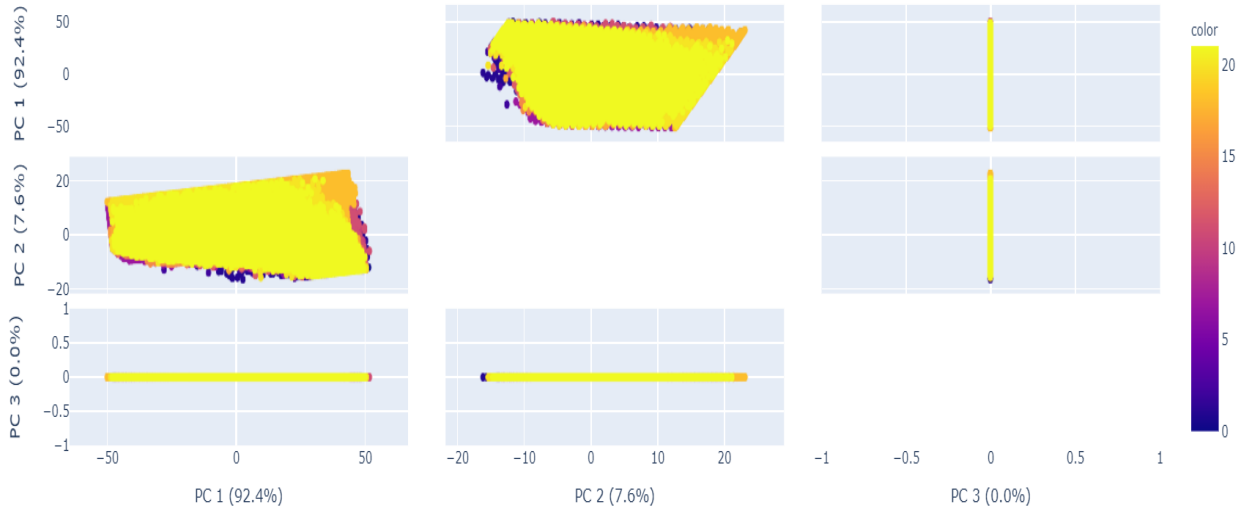



Figure 15: Explained Variance Ration Plot for n_components=3

The below figure shows the PCA generated when number of components is taken as 4. The features taken into account are sun hour, temperature, humidity, and precipitation.

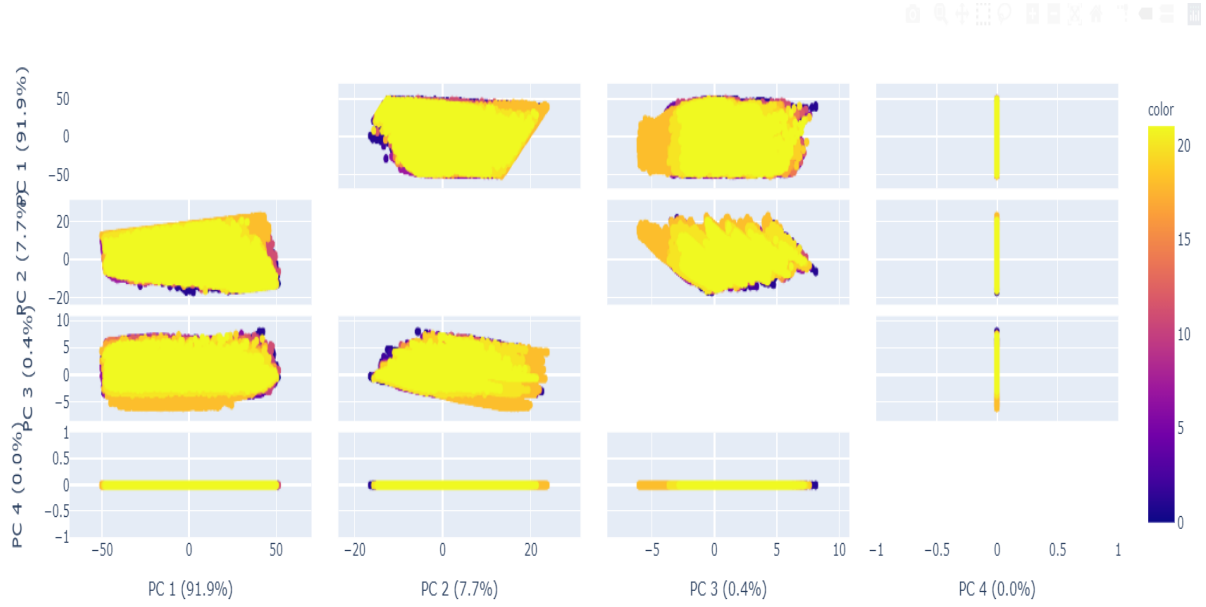



Figure 16: Explained Variance Ration Plot for n_components=4

The below figure shows the PCA generated when number of components is taken as 6. The features taken into account are temperature, humidity, and precipitation.

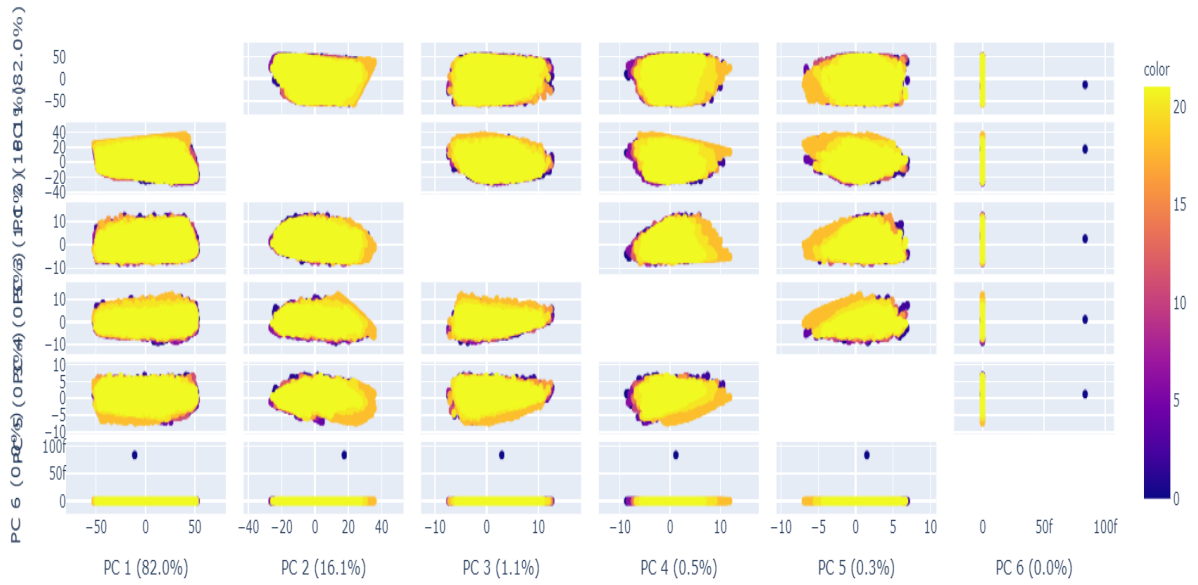



Figure 17: Explained Variance Ration Plot for n_components=6

In the next part of our paper, we implemented prediction of rainfall using Linear Regression, Decision Tree, and Gradient Boosted Decision Tree. As described earlier, the dataset is extracted from a weather data API, which provides the most recent and accurate weather details. Then, a comparative analysis is presented based on the performance of each of these models using RMSE value. Before performing machine learning, we drop the uncorrelated columns such as moonrise, sunrise, moonset, sunset, etc. We split the dataset into training and testing. We conduct this process for each city.

```
In [153]: ahmedabad = ahmedabad.drop(["moonrise",'sunrise',"moonset","sunset"], axis = 1)


          X = ahmedabad.loc[:, ahmedabad.columns != 'precipMM']
          Y = ahmedabad[["precipMM"]]

          X = X.drop(["date_time"], axis = 1)
          X = X.drop(["location"], axis =1)

          from sklearn import datasets, linear_model, metrics

          # Splitting the dataset into the Training set and Test set
          from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

Figure 18: Dropping uncorrelated columns and splitting the dataset

**I) Linear Regression**

The first technique used is the Linear Regression. Linear Regression is a supervised machine learning technique. By performing regression analytics, it predicts the value based on the independent parameters. We import Linear Regression using the command: 'from sklearn.linear_model import LinearRegression'. We use LinearRegression for both our datasets: training and testing. The output of performing the Linear Regression on the trained dataset is the intercept value and coefficients. We

then summarize the model for the training dataset. The RMSE value is the difference between the values predicted by the model and the actual values. The code for performing Linear Regression is as follows:

```
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
# Predicting the Test set results
y_pred = regressor.predict(X_test)

# regression coefficients
print('Coefficients: ', regressor.coef_)
print('Intercept: ', regressor.intercept_)
print("RMSE: %f" % np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Coefficients:  [[-4.78852605e-02  1.38538355e-02 -6.93889390e-18 -4.42435979e-04
  -2.24621300e-02  1.18157640e-03  2.02596219e-02  8.57468463e-02
  -3.98689138e-02 -1.18987652e-01 -9.70679343e-03  8.91618016e-05
  -5.70899052e-03 -6.15550319e-02  5.99399312e-02  3.48262624e-04
  -2.54835425e-04 -1.08269858e-03]]
Intercept:  [64.43407583]
RMSE: 1.753252
```

Figure 19: Code for Linear Regression

**II) Decision Tree Regression**

Decision Tree Regression breaks down a dataset into smaller sets and a Decision Tree is developed. Decision Tree Regression is capable of learning fine details from the training data. Based on the features from the training dataset the model trains itself to learn a series of questions to infer the class labels of the samples. Thus, it makes the decisions based on asking a series of questions. We use the 'from sklearn.tree import DecisionTreeRegressor' function to import Decision Tree Regression. The code to perform Decision Tree Regression is as follows:

```
In [155]: # import the regressor
from sklearn.tree import DecisionTreeRegressor

# create a regressor object
regressor_dt = DecisionTreeRegressor(random_state = 0)

# fit the regressor with X and Y data
regressor_dt.fit(X, y)

y_pred_dt = regressor_dt.predict(X_test)
from sklearn.metrics import mean_squared_error as MSE

# test set RMSE
test_rmse = MSE(y_test, y_pred_dt) ** (1 / 2)

# Print rmse
print('RMSE test set: {:f}'.format(test_rmse))

rms = MSE(y_test, y_pred_dt, squared=False)
print(rms)
```

```
RMSE test set: 0.000000
8.311852437395844e-19
```

Figure 20: Code for Decision Tree Regression

### III) Gradient Boosted Decision Tree Regression

Gradient Boosted Decision Tree Regression is the machine learning model used for the regression. The main idea of Gradient Boosted Decision Tree Regression is to produce a prediction model by the ensemble of weak prediction models. We import GBT regression using the 'from sklearn.ensemble import GradientBoostingRegressor'. The code to perform Gradient Boosted Decision Tree Regression is as follows:

```
In [157]: from sklearn.ensemble import GradientBoostingRegressor

          # Instantiate Gradient Boosting Regressor
          gbr = GradientBoostingRegressor(n_estimators = 200, max_depth = 1, random_state = 1)

          # Fit to training set
          gbr.fit(X_train, y_train)

          # Predict on test set
          pred_y = gbr.predict(X_test)

          from sklearn.metrics import mean_squared_error as MSE

          # test set RMSE
          test_rmse = MSE(y_test, pred_y) ** (1 / 2)

          # Print rmse
          print('RMSE test set: {:.2f}'.format(test_rmse))

          C:\Anaconda\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning:
          ay was expected. Please change the shape of y to (n_samples, ), for example using rav
            return f(**kwargs)

          RMSE test set: 1.74
```

Figure 21: Code for Gradient Boosted Decision Tree Regression

This process is repeated for each city. Each of the three models offers a RMSE value and that value is recorded for the comparative analysis. The summarized information is displayed below in Table 1. The table lists the RMSE values of each model.

| Cities | Linear Regression | Decision Tree | Gradient Boosted Decision Tree |
|---|---|---|---|
| agra | 1.383687 | 6.84E-18 | 1.18 |
| ahmedabad | 1.753252 | 8.31E-19 | 1.74 |
| amritsar | 1.732916 | 1.95E-18 | 1.72 |
| bengaluru | 1.770868 | 1.21E-18 | 1.76 |
| chandigarh | 1.725416 | 9.96E-19 | 1.71 |
| chennai | 1.770435 | 2.43E-18 | 1.76 |
| delhi | 1.688762 | 1.68E-18 | 1.66 |
| guwahati | 1.768467 | 1.41E-18 | 1.76 |
| hyderabad | 1.759865 | 8.73E-19 | 1.75 |
| indore | 1.741683 | 3.10E-18 | 1.73 |
| jaipur | 1.705292 | 9.12E-19 | 1.69 |
| jodhpur | 1.740001 | 1.99E-18 | 1.73 |
| kolkata | 1.763412 | 1.38E-18 | 1.75 |
| kota | 1.787547 | 5.15E-19 | 1.78 |
| lucknow | 1.714705 | 9.87E-19 | 1.69 |
| mumbai | 1.752638 | 1.04E-18 | 1.74 |
| pune | 1.74947 | 1.42E-18 | 1.75 |
| ranchi | 1.753741 | 1.80E-18 | 1.74 |
| srinagar | 1.75684 | 3.07E-18 | 1.74 |
| surat | 1.749796 | 1.06E-18 | 1.74 |
| thane | 1.717498 | 7.76E-19 | 1.7 |
| varanasi | 1.735926 | 8.73E-19 | 1.72 |

Table 1: RMSE Value Comparison for Each Model in Each City

As we can conclude, Decision Tree Regression performs the best as it provides the least RMSE value. With the exception of one city (Pune), Gradient Boosted Decision Tree offers the second-best result.

The nest part of our paper deals with classification of rainfall levels.

The target data is a real number and hence binned into the following classes. Now the problem can be converted to a classification problem and sliding window can be applied as a hyperparameter to solve the problem.

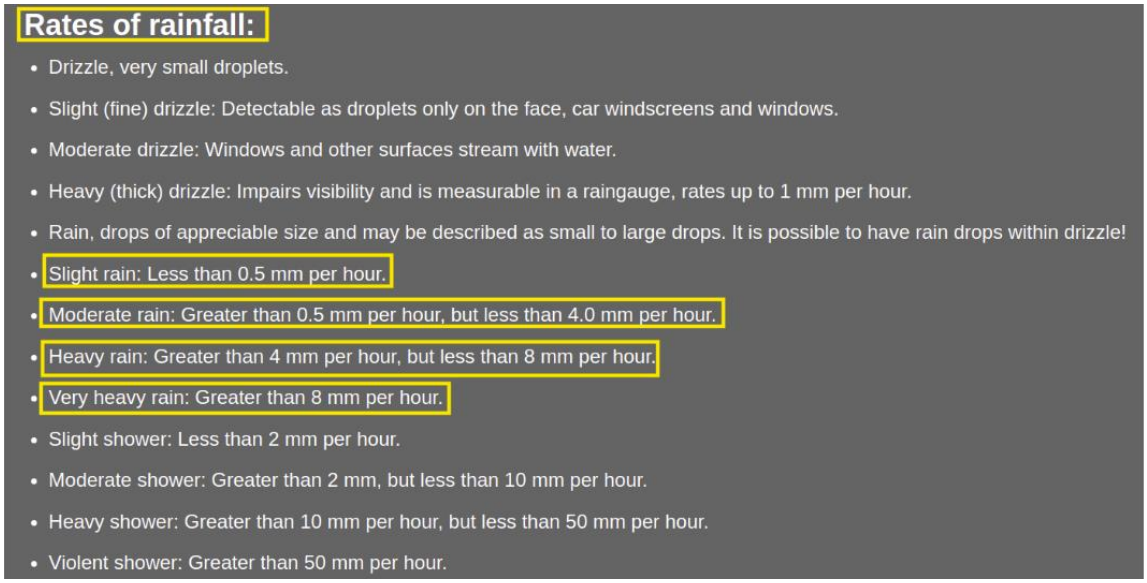



Figure 22: Rainfall Classification Levels

The following is the Percentile plot for the dimension maxtemp for multiple cities.

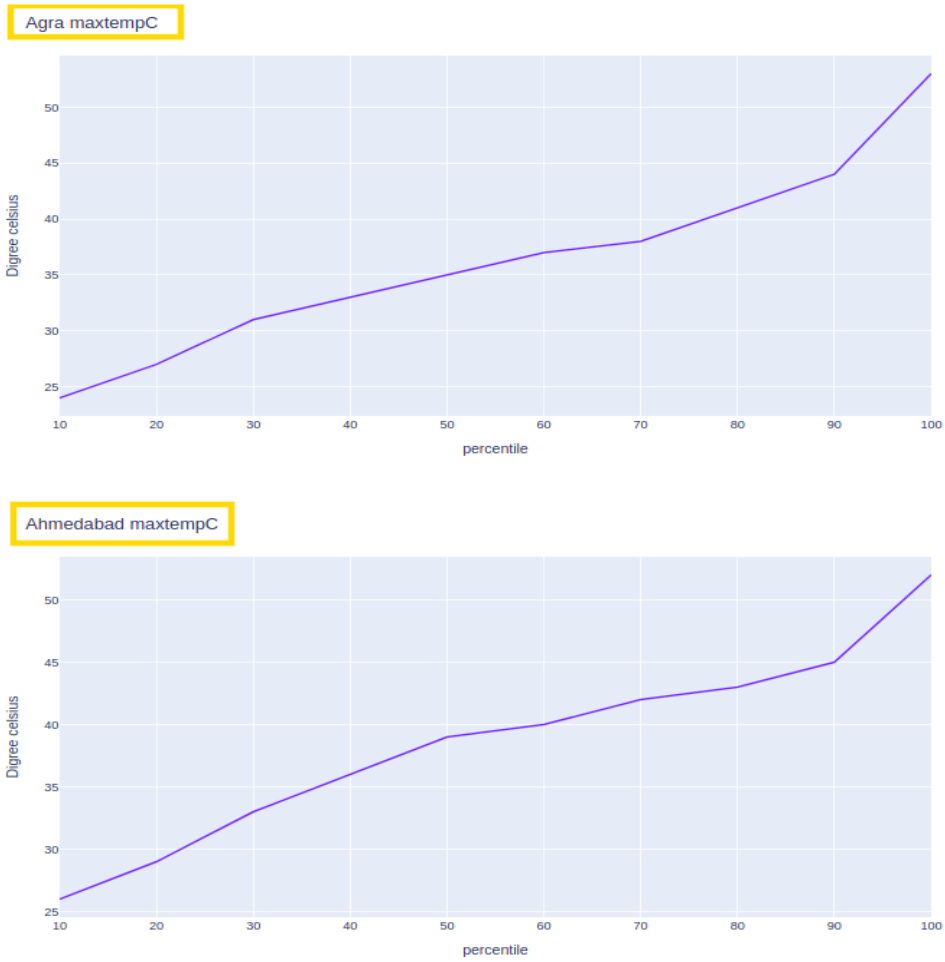



Figure 23: Percentile plot for the dimension maxtemp

The following is the percentile plot for the target column precipMM.


Figure 24: Percentile plot for the target column

The following is the distribution data in different class.

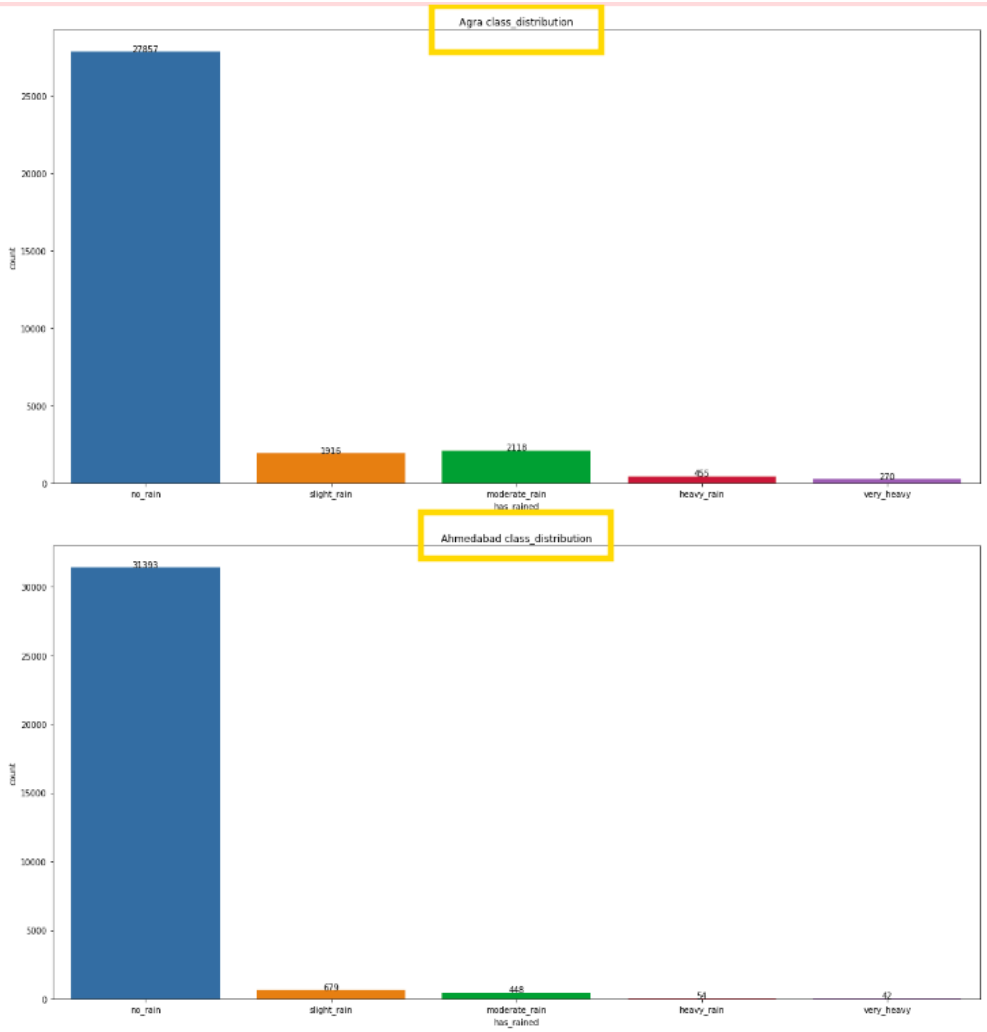



Figure 25: Distribution data

The following is the time series plot for column max_temp_plot

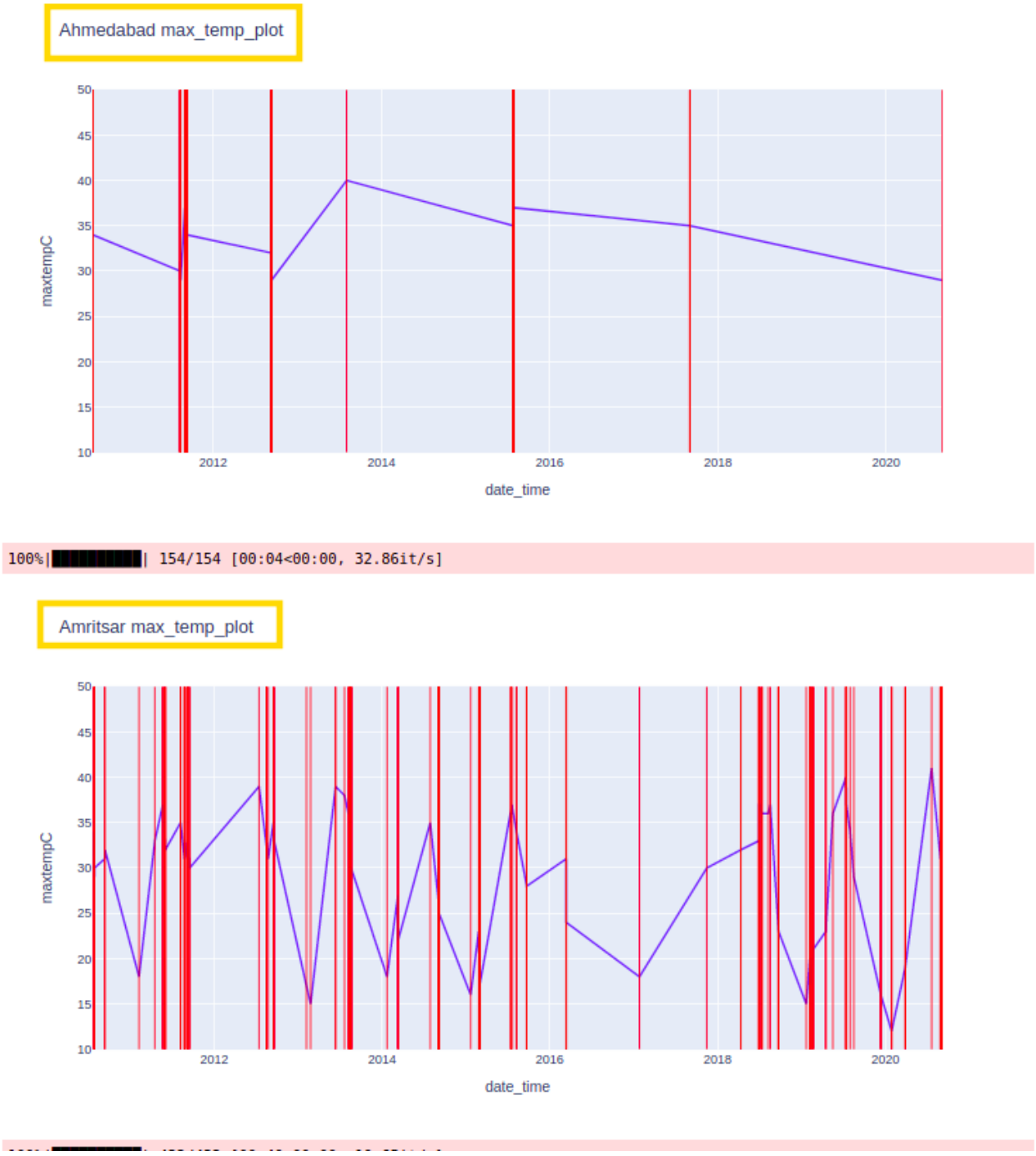



Figure 26: time series plot for column max_temp_plot

The tsne Plot is in the following.

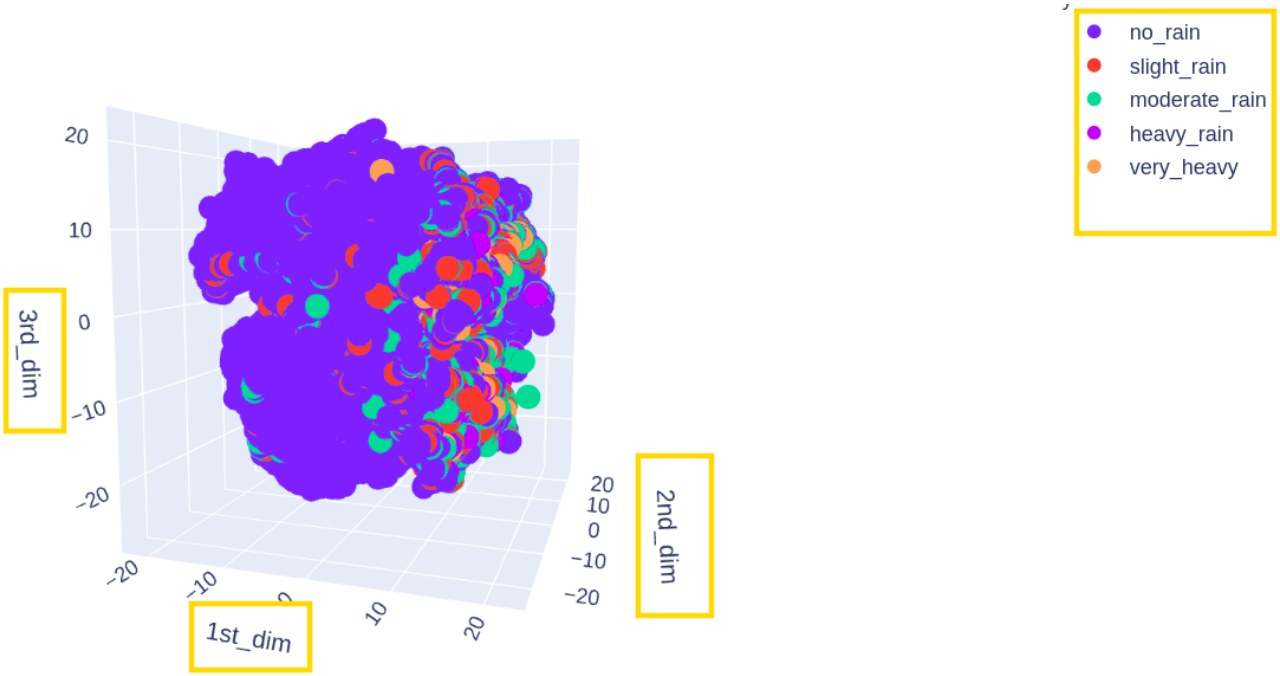



Figure 27: t-SNE plot for rainfall levels

The following is the script for sliding window.

```
In [163]: for_tsne.shape
Out[163]: (717552, 19)

In [127]: def window_stack(a, stepsize, width):
              n = a.shape[0]
              return np.hstack(a[i:1+n+i-width:stepsize] for i in range(0,width))

In [128]: windowed=window_stack(np.array(for_tsne[0:50000]),1,3)

In [129]: windowed.shape

Out[129]: (49998, 57)

In [130]: y=y[2:]

In [131]: y.shape
Out[131]: (49998,)
```

Result of model logistic regression using sliding window of size 3

```
Predicting test data
Done


testing time(HH:MM:SS:ms) - 0:00:00.011037


---------------------
|      Accuracy      |
---------------------

    0.8955151515151515

--------------------
| Confusion Matrix |
--------------------

 [[    5    73    79     2     1]
 [    8   225   520    14     7]
 [    2   130 14527     7     2]
 [    7   145   631     5     3]
 [    6    51    36     0    14]]
```

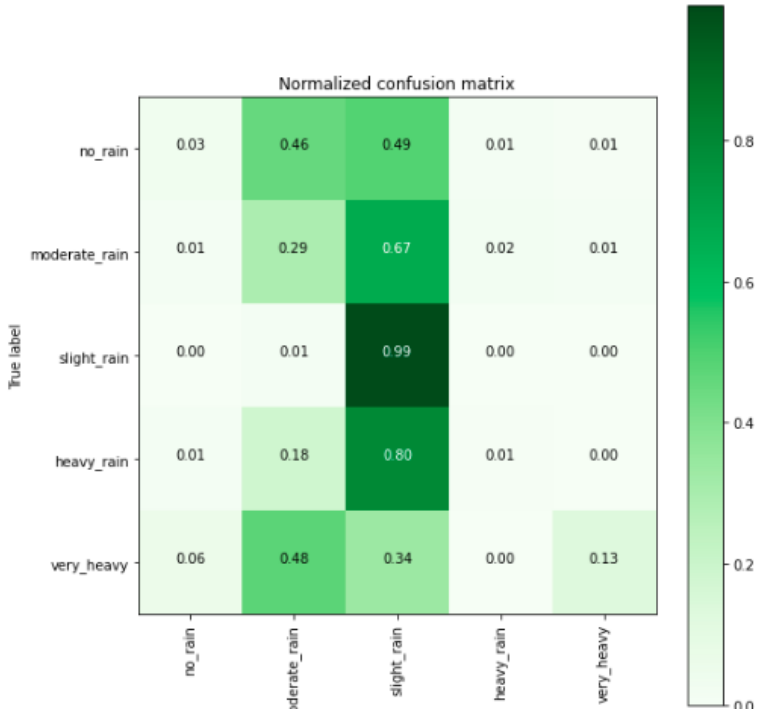



Figure 28: Result of logistic regression

Result of model linear SVM sliding window of size 3

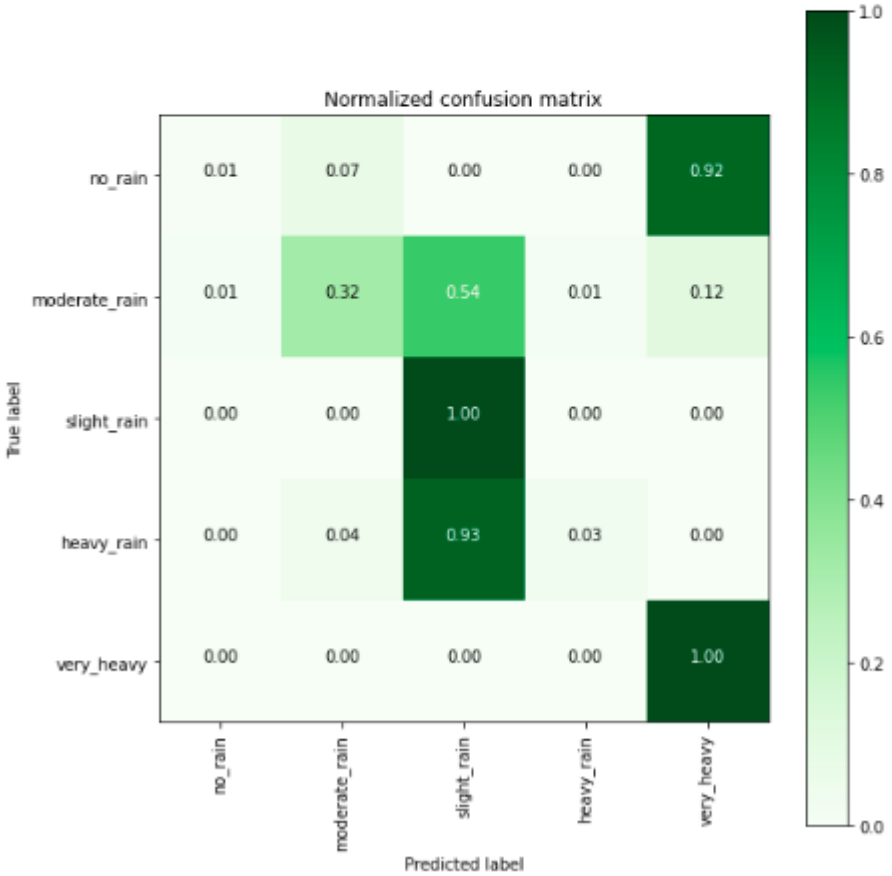
```
testing time(HH:MM:SS:ms) - 0:00:00.002325

---------------------
|      Accuracy      |
---------------------

    0.9096363636363637

--------------------
| Confusion Matrix |
--------------------

 [[    1    12     0     0   147]
 [    8   246   420     8    92]
 [    0    19 14635     9     5]
 [    0    32   738    20     1]
 [    0     0     0     0   107]]
```




```
--------------------------
```

Figure 29: Result of linear SVM

Result of model decision tree using sliding window of size 3

```
training_time(HH:MM:SS.ms) - 0:00:01.250320

Predicting test data
Done

testing time(HH:MM:SS:ms) - 0:00:00.002695

---------------------
|      Accuracy      |
---------------------

    1.0

--------------------
| Confusion Matrix |
--------------------

 [[  160     0     0     0     0]
 [    0   774     0     0     0]
 [    0     0 14668     0     0]
 [    0     0     0   791     0]
 [    0     0     0     0   107]]
```

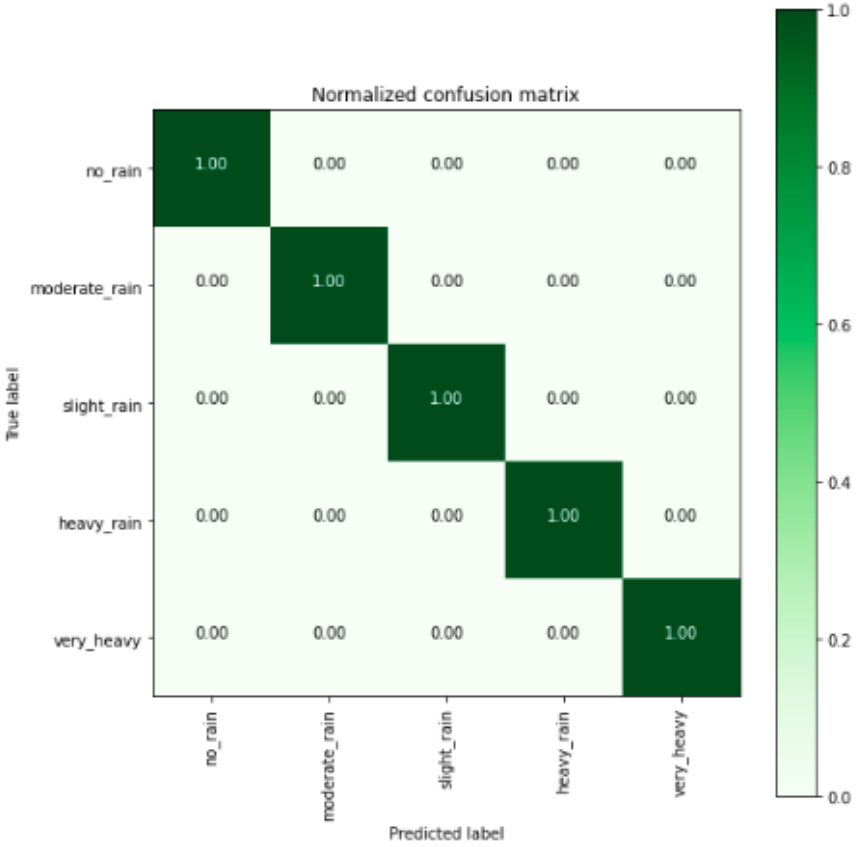



Figure 30: Result of decision tree

Result of model random forest using sliding window of size 3

```
Predicting test data
Done

testing time(HH:MM:SS:ms) - 0:00:00.534856

---------------------
|      Accuracy      |
---------------------

    0.9897575757575757

--------------------
| Confusion Matrix |
--------------------

 [[   54   101     0     0     5]
 [    0   753    15     6     0]
 [    0     0 14668     0     0]
 [    0    13     0   778     0]
 [    3    26     0     0    78]]
```

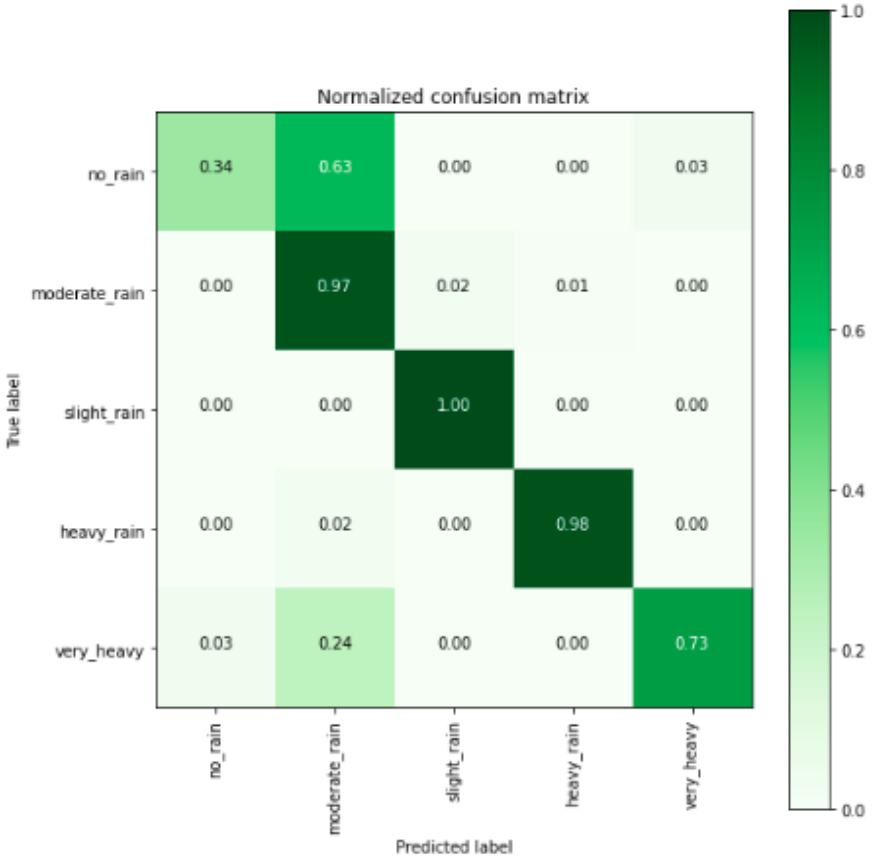



Figure 31: Result of random forest

Comparison of models is summarized below:

| Model | accuracy |
|---|---|
| logistic regresssion | 89.551 |
| linear svm | 90.963 |
| Decision Trees | 100 |
| Random Forest | 98.975 |

Table 2: Comparison of Models in Classification

As we can conclude, decision tree and random forest algorithms offer the most accuracy.

## Conclusion

Natural calamities like floods and landslides cause significant human losses and serious harm to physical, organizational structures and facilities and natural resources. Preventing this phenomenon to occur is beyond the scope of human power. The best thing to do is to minimize the impact by predicting before the occurrence of such calamities. These happenings are generally linked with periods of continuous rainfall or otherwise rapid snowmelt making rainfall prediction an efficient technique for predicting the occurrence of these calamities. Our paper provides a method for predicting rainfall and classifying it. The time series analysis on weather data is performed and rainfall prediction analysis is done on the weather data with the help of machine learning algorithms like Linear Regression, Decision tree regression and Gradient Boosted regression. Among them, decision tree regression gave better accuracy compared to other models. Similarly, from the classification models like Support Vector Machine (SVM), Logistic Regression, Decision tree and Random Forest used for classifying the rainfall into different levels, Decision tree and Random Forest gave better results in terms of accuracy.

## References

[1] L. R. Varghese and K. Vanitha, "A Time-series based Prediction Analysis of Rainfall Detection," 2020 International Conference on Inventive Computation Technologies (ICICT), 2020, pp. 513-518, doi: 10.1109/ICICT48043.2020.9112488.

[2] U. Ashwini, K. Kalaivani, K. Ulagapriya and A. Saritha, "Time Series Analysis based Tamilnadu Monsoon Rainfall Prediction using Seasonal ARIMA," 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 1293-1297, doi: 10.1109/ICICT50816.2021.9358615.

[3] R. K. Grace and B. Suganya, "Machine Learning based Rainfall Prediction," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 227-229, doi: 10.1109/ICACCS48705.2020.9074233.

[4] S. K. Mohapatra, A. Upadhyay and C. Gola, "Rainfall prediction based on 100 years of meteorological data," 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), 2017, pp. 162-166, doi: 10.1109/IC3TSN.2017.8284469.

[5] Y. K. Joshi, U. Chawla and S. Shukla, "Rainfall Prediction Using Data Visualisation Techniques," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2020, pp. 327-331, doi: 10.1109/Confluence47617.2020.9057928.