UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

2022

Big Data (6CS030)

Course Report on:

Big data analytics on Weather Forecasting/Prediction impact of business and daily life

Student Id	: 2050214
Student Name	: Yogesh Shrestha
Group	: L6CG8
Module Leader	: Jnaneshwar Bohara
Submitted on	: 02-05-2022

Table of Contents

Table of Figures

Abstracts

This research shows, the weather prediction of the data using various models like, decision tree, logistic regression, and random forest regression. The weather predication's data contain 8 columns which "weather column" is target variable. This research has cover up about how big data is analyzed, how search engine (elastic search) is used to visualize, how Jupiter notebook and its libraries is used, and how Apache spark is used. It also talks about why do we clean data? Why do we visualize the data? Why do we use models? While predicting weather this research has used libraries NumPy and pandas to read and operate data, matplotlib, seaborn to visualize data, missing to find missing values, LablelEncoding to convert categorical to numerical values etc. the 3 model were used and hyper tuning is applied to find the best modes for the prediction. Also used elastic search engine to visualize the data in simple way. And also, used spark system to analyze data using queries.

Background of Weather Forecasting

Big data is the data which contains the all type of data such as relational data, non-relational data and undefined data also that means unshaped data in any kind of formation which is quite difficult to measure these kinds of data and define the own mechanism to predict their future. Data mining is the option to process the weather data, calculate certain mechanism and predict the value forecasting using the machine learning. Nowadays weather furcating is most import part of our life because of its effect on several areas like business, human life, agriculture and other factors also. (Shubham Madan, 2018)

Today's world weather forecasting plays the vital role which can be solved many problems which occurs from the weather condition. Nowadays, there are multiple sectors which can be affect from the weather changes which can be harmful for our human lives, business losses, farming and agriculture and other factor also. (A. K. Pandey, 2017)

The main objective of the weather forecasting is that it includes carries the essentials information such as temperature, condition of weather and predictions. Rainfall, humidity etc. which can be effect of atmospheric data. These given data used to prevent climate hazard. (A. K. Pandey, 2017)

By using the data mining the unstructured data are easily access the from the dataset where it can be helpful for the future prediction on weather forecasting to solve our weather-related losses and daily life. Data mining play the vital role to manipulate the data information to build the relation and predict from data. By using the methodology several method and algorithms are available but in the case of data mining or big data with help of machine learning basically focused on Regression and Classification along with Clustering and the Associations. (Shubham Madan, 2018)

The concept of Big Data may be conveyed by focusing on the large data definition, the information order and wellsprings of enormous data, as well as its prominent qualities, data sets, and cycles. According to Big Data: The Next Frontier for Innovation, there are five hierarchical sectors for huge data to create esteem in light of the size of the potential, including medical care, manufacturing, public sector organization, retail, and global individual area data. There are several potential linkages that need massive information configurations, such as logical revelation (for example, cosmic associations, climatic predictions) and a huge amount of information. (Pwint Phyu Khine, 2017)

Related Work

Design of Short-Term Wind Production Forecasting Model using Machine Learning Algorithms

In this research paper, the wind power production forecast is analyzed with various approach model of the machine learning (ML). it uses python, python', required libraries, and python logics in ML approaches. The collected data are analyzed, visualized, cleaned, used feature selection, pre-processed data, normalized data, trained the data with various approached, and used hyper-parameter tuning method to find best model. To forecast the wind production, this model is created using Python's pandas (for data analysis), matplotlib (for visualization), seas born (for visualization) and Scikit-learn (for classification, clustering, random forest, k-means, etc.). The suggested ML approach for WPP operators to use in wind energy performance estimations is mentioned in the article.

The data is pre-processed in order to expand the best accuracy possibilities of wind energy suppliers on methodologies platforms. For data preprocessing te following equation is used i.e.

x` = (x - xmin) / (xmax - xmin) where: x is value of normalized, xmin is the minimum and xmax is the maximum value.

The essential algorithms like Linear regression, Polynomial Ridge regression, Decision Tree regression, Gradient Boosting regression etc. were used to build this model. For linear regression following equation is used i.e.

Y = b0 + b1. x + error   where: y is dependent variable, b0 is constant term, b1 is coefficient of independent variable.

The best method was selected i.e., Polynomial Ridge Regression for accurate prediction. While predicting using the model there was some predictive error which was solved using MAE, MAPE, NMAE, RMSE. The error was handled using following equations i.e.

Mean Absolute Error (MAE)
$$= \frac{\sum_{i=1}^{n} |y_{pred} - y_{real}|}{n}$$

Mean Absolute Percentage Error (MAPE)
$$= \frac{100}{n} \frac{\sum_{i=1}^{n} |y_{pred} - y_{real}|}{y_{real}}$$

Normalized Mean Absolute Error (NMAE)
$$= \frac{avg \sum_{i=1}^{n} |y_{pred} - y_{real}|}{y_{real\,max} - y_{real\,min}}$$

Root Mean Square Error (RMSE)
$$= \sqrt{\sum_{i=1}^{n} (y\_real - y\_pred)^2}$$

Where: n is the numbers of samples, real is real produced power, Ypres is forecast wind energy generation. (Eugen, 2021)

Just like wind production forecasting, our weather can be predicted using several algorithms like linear regression, polynomial ridge regression, decision tree etc. to build models. We can use the visualization technique of matplotlib, seaborn to visualize, scikit-learn for classification, data and pre-process data by normalizing it for better accuracy. Then, select the best model for our weather prediction using hyper parameters and calculating error using MAE, RMSE etc. (Marius Eugen ȚIBOACĂ, 2021)

Big Data prediction framework for weather Temperature based on MapReduce algorithm

In this research paper, it focused on a MapReduce for the Big Data prediction models of weather temperature. It has solved the problem of gathering high volume data and velocity of sensors data in the process of gathering data. A vast amount of data was collected utilizing a scalable analytical approach with the aid of big data. Inside Big data there is an architecture named “Hadoop’s MapReduce” which help to collect sessors data scalable without data limitations. Hadoop is known as the Big Data analytics platform, whereas MapReduce is known as the processing engine at its core.

While building this project Hadoop framework were used which helps in collecting and analyzing the vast volumes of the data. The collected and analyzed data process were both efficient and effective. And MapReduce were used for taking all collected data and convert it into new set of data and then, get map’s result as output. The Hadoop Distributed file system were used to analyze and handle cluster data. With minimum 1 mapper, the input data contains columns for temperature, time, and location that were separated into data blocks based on its block size. The mapper script runs on the Data Node and retrieves and analyzes every separated block of data, referred to as a "record." The filtered outputs are then given to Lower tasks, which reduce the data volume's findings and calculating the average temperature using the mapper algorithm. The columns are then integrated, or decreased, to get the end results. The use of these techniques to interpret data has the potential to improve weather prediction. (Ismail, 2016)

Just like weather temperature prediction, to analyze weather data as Big Data, MapReduce using Hadoop is an excellent choice. As we know, we use MapReduce framework for employing a large number of commodities devices to run massively parallel processing and distributable algorithms over enormous datasets which helps to properly analyze. We can use this to fix scalability issue as well. (Ismail, 2016)

Methodology

Jupyter notebook are open-source web apps that act as virtual lab note pads, allowing users to analyze datasets, create perceptions, display computations, and much more. A basic JSON journal, the server-client application, and a note pad part that runs the code in the JSON scratch pad make up the anaconda environment. JNs may be used by an analyst with a lot of flexibility because of the underlying simplicity. Jupyter notebook, previously known as iPython journals, were intended to provide faster, "on the fly" processing for information exploration, saving significant time and effort over traditional methods of dealing with ordering code. However, as Jupyter notebook were more widely used, experts began to use them to aid in all computational phases of the examination lifecycle, including data research, dissemination, and training. (Wofford, 2019)

MongoDB is a C++-based data set designed for scattered document capacity. It is a non-social data source. MongoDB is frequently used to store unusually large amounts of data. Using MongoDB is primarily a process of breaking data into parts and then storing them on separate devices. As though it were a separate MongoDB, the data set design is unnoticeable to the application. We primarily built up a MongoDB group to work with. (Lu Han, 2020)

A fraction of Elasticsearch's phrasing and thoughts need be dominated in order to make sense of it. One of the most important concepts in Elasticsearch is lists. The data in ES is saved in a record, and we get the data from the list. A sort is a consistent parcel inside a record that can be classified into at least one kind. JSON design addresses archives, which are the fundamental unit of ordering and looking. Every ES bunch is made up of several hubs, and each running example in ES is a hub. The information in a list is separated into shards. A shard is a finalized web index that is similar to a Lucene case. Clustering is a data storage compartment where records are stored. Implementation is subsequently assigned to each of the cluster's hubs. (Lu Han, 2020)

Apache Spark was created for increased speed, usability, and in-depth exploration. The MLlib library, which is used for AI approaches, is included with Apache Spark. Apache Spark works well with Java, Python, R, and Scala designers because it offers a large number of APIs and a practical approach. In Spark, a single library may do a variety of tasks, such as SQL, streaming, and graphical analysis. Unlike other approaches such as Hadoop Map-Reduce, Spark uses in-memory calculations to process data quickly and effectively. It can handle massive amounts of data in the terabytes and petabytes range at the same time and conduct excellent data management. (Comparative Analysis of Hadoop Tools and Spark Technology, 2018)

Logistic Regression

It is all about categorial data representation when data has only discrete value other than labeled value for example in this project rainfall probability is predict either it is rain or not. This prediction is given by below equation. (Sharma, 2019)

The fundamental equation of generalized linear model is:

$$g(E(y)) = \theta_0 + x_1\theta_1 + x_2\theta_2 + x_3\theta_3 + x_4\theta_4 + x_5\theta_5 + \cdots$$

Logistic regression is always depended on the value of theta, which is also called the cost function. Let us discuss about above formula g () function is describe as link function and E(y) is the targeted value which is also called the dependent value. In logistic regression (p) stand for the probability of success and the failure is about difference between 1 to success i.e. (1-p). (Sharma, 2019)

simply be done by:

$$L(x) = \frac{\exp(g(E(y)))}{1+\exp(g(E(y)))} \qquad (2)$$

Using the above formula L(x) can be written as:

$$L(x) = \frac{e^y}{e^1 + e^y} \qquad (3)$$

where,

$$g(E(y)) = \theta_0 + x_1\theta_1 + x_2\theta_2 + x_3\theta_3 + x_4\theta_4 + x_5\theta_5$$

Random Forest Classifier

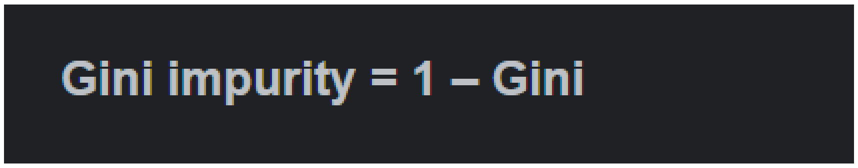
This technique mainly focused on the non -linear statical analysis which is diagnose the atmospheric temperature of the given region that assured to the maximum and minimum temperature which may cause of calamite hazard. By using the random forest in the weather dataset with various time year data can be collected to compare the variable. These algorithms basically analyze the all-over entire days temperature which also most important for stability and sensible for the future weather predictions. (John K. Williams* and D. A. Ahijevych, 2020)

Decision tree

One of the most common algorithms for data retrieving to predict the target value according to their various input parameters. It is split based learning system that divided into sub part of the main part with the recursive functionality and it follow the top-down approach to train and predict the data. (Kumar, 2022)

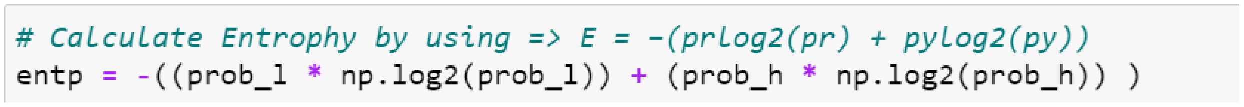
To differentiate weather's data, use the Gini pollutant. The Gini pollutant is a fraction of the likelihood that a randomly selected component from a collection would be incorrectly labeled if it were arbitrarily called by the dispersion of names in the subset. We define pj as the generic recurrence of class j in a collection D comprising models from n classes. (Zhi, 2018)

It calculates Gini index at first as given below,


Gini impurity = 1 – Gini

Entropy

The entropy of the closed and a few varied frameworks grows with time, eventually covering all of the areas of the framework's conceivable states and motions. On the other hand, it has been noted that the collection of data included in a logical hypothesis outnumbers the assemblage of data contained in rudimentary experimental data. Each of



```
# Calculate Entrophy by using => E = -(prlog2(pr) + pylog2(py))
entp = -((prob_l * np.log2(prob_l)) + (prob_h * np.log2(prob_h)) )
```

the realities has its own explanation. For example, logical laws are devised by the human creative mind and discovered by a human, despite the fact that they are based on experimental observations. As a result, the outcomes of innovative labor and information about a researcher are the source of surplus data. However, despite the current position of the data hypothesis and negentropy standard allowing for this, the facts raised do not have a satisfactory numerical elucidation. (Chernyshov, 2005)

Block Diagram

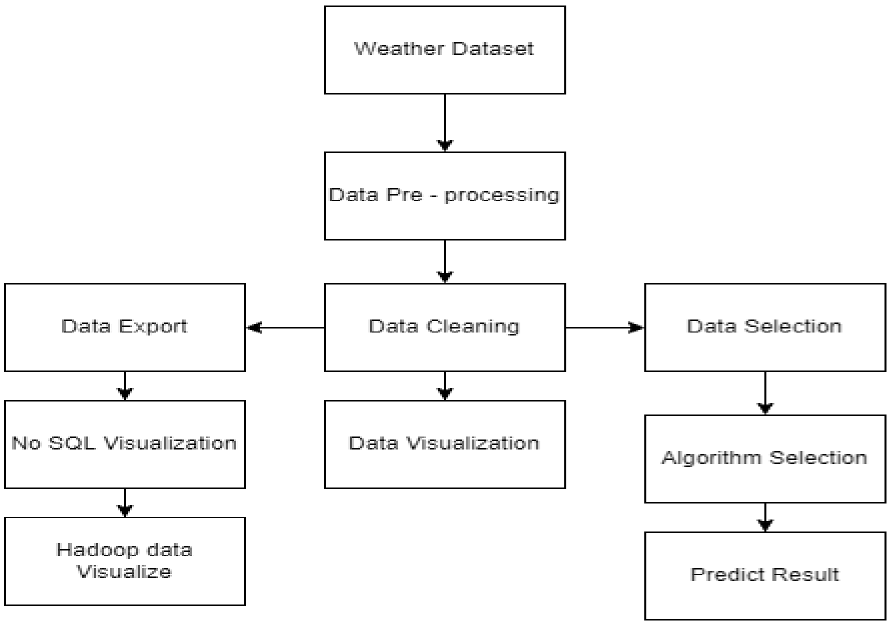


Figure 1:block Diagram of weather prediction

In this figure, it shows the block diagram of weather prediction process.

Block Diagram Process

Weather Dataset

In this weather data, it contains 8 columns in which weather column is used as target variable and remaining was used for independent variables.

Data Pre-processing

In the data pre-processing, the NAN value was searched and cleaned. The categorical value was converted to numerical value so that accuracy of data increase.

Data cleaning

In the process of data cleaning, the outlier was removed, the nan value was filled with average value of the median.

Data Visualization

The data visualization was using juypter notebook's library such as: matplotlib, seaborn, histogram. The analytical engine named "elastic search" was also used for the data visualization.

Data Selection

The required data was select for the prediction of the models. The unnecessary data, which does not effect in prediction process is dropped.

Algorithm Selection

After training 3 models i.e. decision tree, logistic regression, and random forest regression the best model was selected using hyper parameter tuning.

Predict Result

After selecting best model, the prediction showed 70% of accuracy which was logistic regression.

Data Export

After cleaning data, exporting data.

No SQL Visualization

Using NO SQL visualization, to visualize the data.

Hadoop spark data Visualization

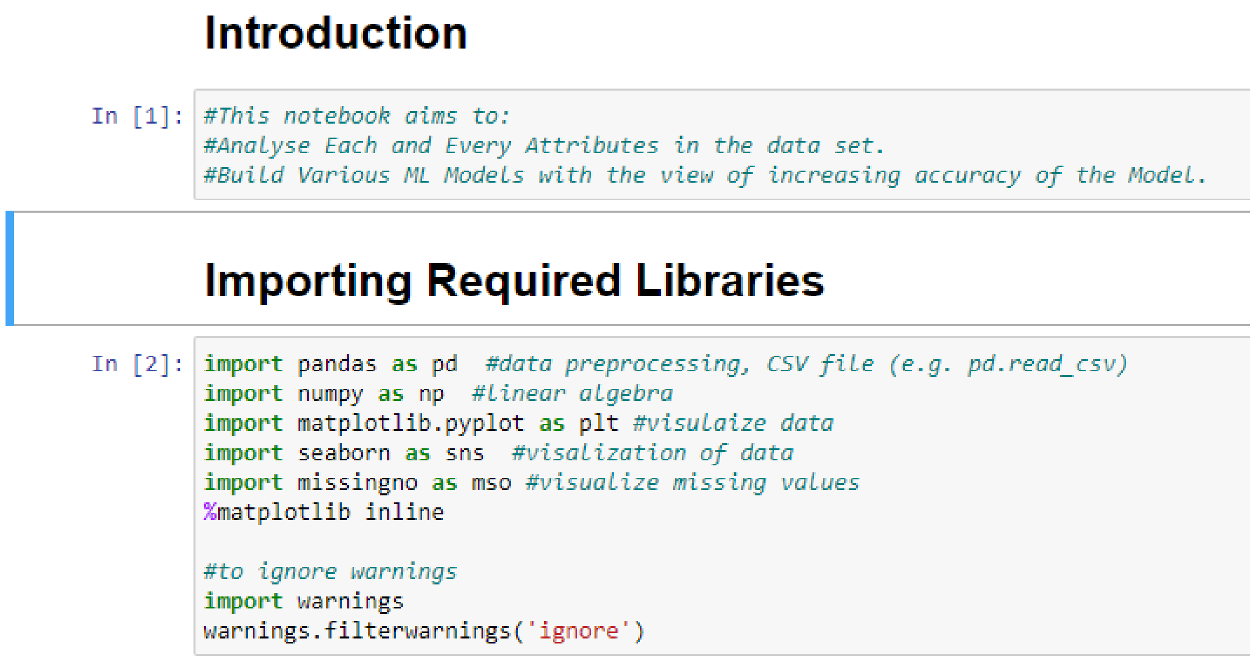
Using Hadoop spark, the data was analyzed using filter queries.

Result and Discussion

The data contains 8 columns i.e., date, precipitation, temp_max, temp_min, wind, summary, wind_direction and weather. Whereas, weather is the target variable. the data holds the record of data. For example, if there is high precipitation, hight temp_max, low temp_min, average wind rate, weather type is party cloud and direction is coming form north then there is high chance of having rainy day.

In the process of weather prediction, the following tools and technology were used i.e., Jupyter notebook, spark, Elastic search, NOSQL etc.

Using Jupyter Notebook



Introduction

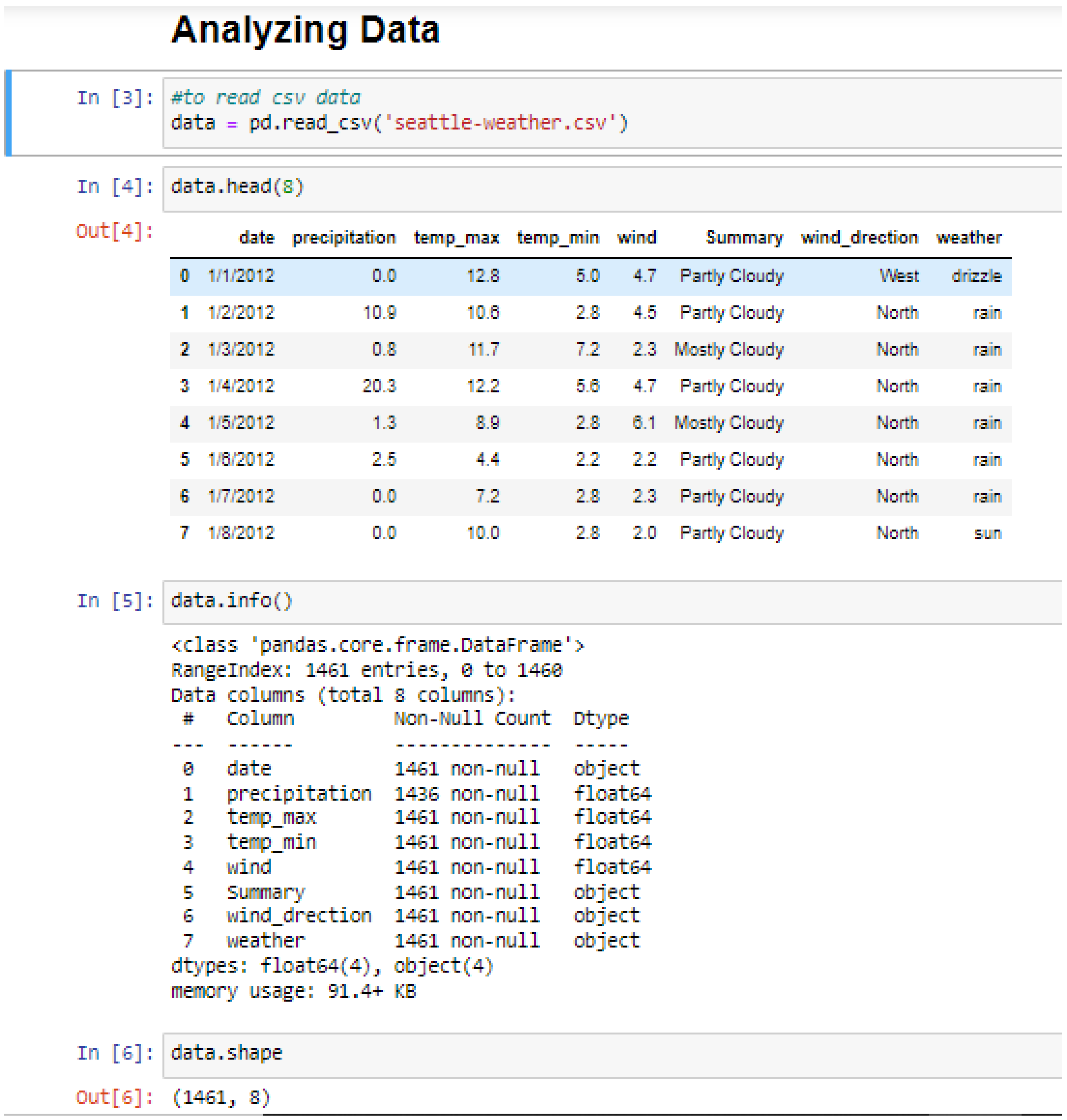
```
In [1]: #This notebook aims to:
        #Analyse Each and Every Attributes in the data set.
        #Build Various ML Models with the view of increasing accuracy of the Model.
```

Importing Required Libraries

```
In [2]: import pandas as pd  #data preprocessing, CSV file (e.g. pd.read_csv)
        import numpy as np  #linear algebra
        import matplotlib.pyplot as plt #visulaize data
        import seaborn as sns  #visalization of data
        import missingno as mso #visualize missing values
        %matplotlib inline

        #to ignore warnings
        import warnings
        warnings.filterwarnings('ignore')
```

Figure 2: importing libraries

Importing necessary libraries like pandas, numpy, matplotlib.pylot, seaborn and so on. These libraries help to clean, process and visualize the data.



Analyzing Data

```
In [3]: #to read csv data
        data = pd.read_csv('seattle-weather.csv')
```

```
In [4]: data.head(8)
```

Out[4]:

	date	precipitation	temp_max	temp_min	wind	Summary	wind_drection	weather
0	1/1/2012	0.0	12.8	5.0	4.7	Partly Cloudy	West	drizzle
1	1/2/2012	10.9	10.6	2.8	4.5	Partly Cloudy	North	rain
2	1/3/2012	0.8	11.7	7.2	2.3	Mostly Cloudy	North	rain
3	1/4/2012	20.3	12.2	5.6	4.7	Partly Cloudy	North	rain
4	1/5/2012	1.3	8.9	2.8	6.1	Mostly Cloudy	North	rain
5	1/6/2012	2.5	4.4	2.2	2.2	Partly Cloudy	North	rain
6	1/7/2012	0.0	7.2	2.8	2.3	Partly Cloudy	North	rain
7	1/8/2012	0.0	10.0	2.8	2.0	Partly Cloudy	North	sun

```
In [5]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           1461 non-null   object
 1   precipitation  1436 non-null   float64
 2   temp_max       1461 non-null   float64
 3   temp_min       1461 non-null   float64
 4   wind           1461 non-null   float64
 5   Summary        1461 non-null   object
 6   wind_drection  1461 non-null   object
 7   weather        1461 non-null   object
dtypes: float64(4), object(4)
memory usage: 91.4+ KB
```

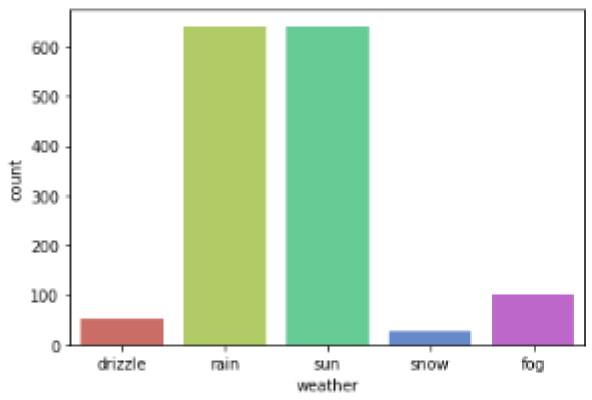
```
In [6]: data.shape
Out[6]: (1461, 8)
```

Figure 3: Analyzing data

Reading csv files using panda. Analyzing the data have 8 columns and each column have either object value or float value. There are 1461 rows of data.

Data Exploration

```
In [7]: sns.countplot("weather",data=data,palette="hls")
Out[7]: <AxesSubplot:xlabel='weather', ylabel='count'>
```



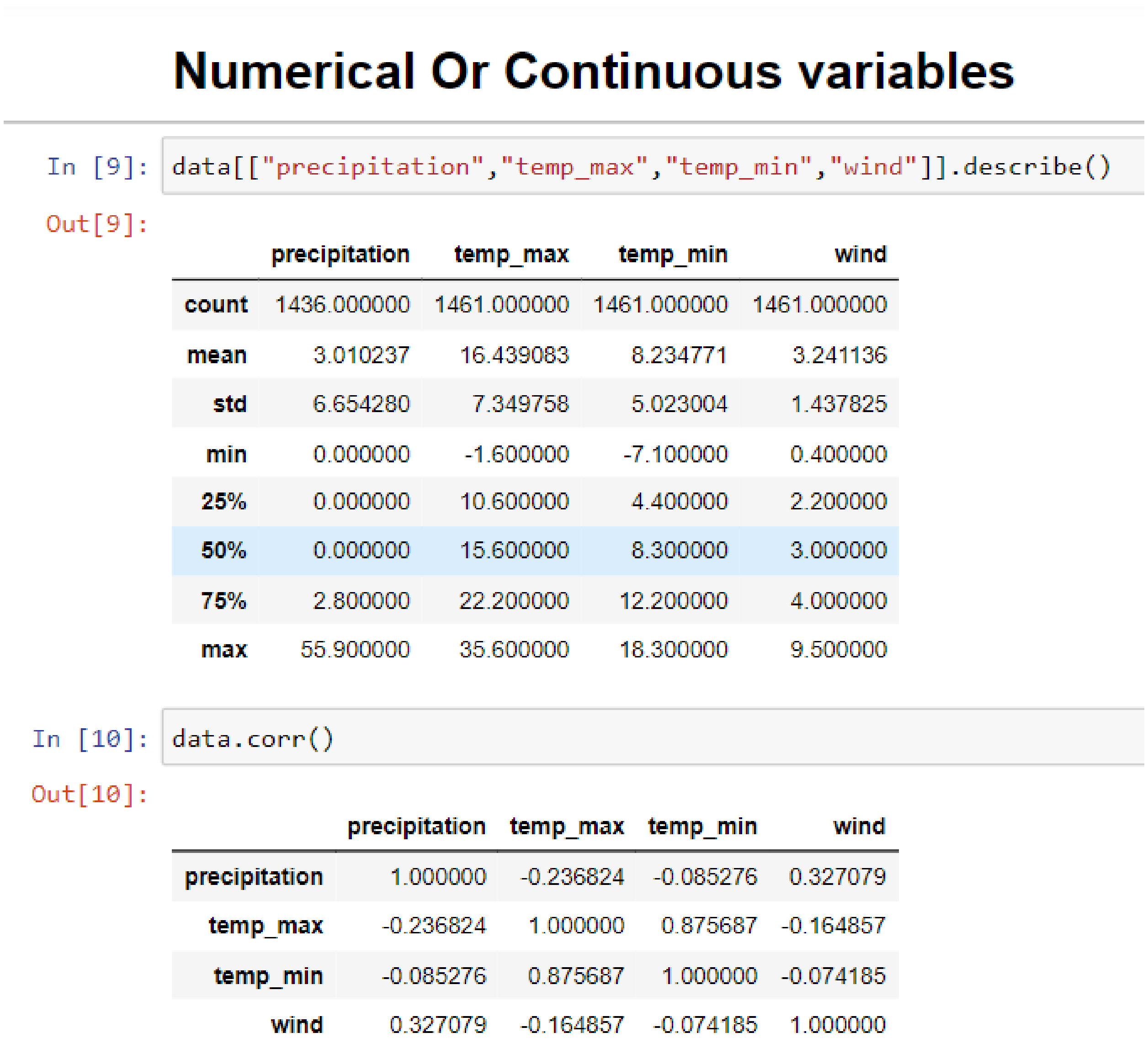
this data shows, the sum of drizzle, rain, sun weather, snow,and fog days.

whereas, ranin and sun weather day ahs highest count but snow has lowest count.

```
In [8]: #Displaying percentage of day's data
        countrain=len(data[data.weather=="rain"])
        countsun=len(data[data.weather=="sun"])
        countdrizzle=len(data[data.weather=="drizzle"])
        countsnow=len(data[data.weather=="snow"])
        countfog=len(data[data.weather=="fog"])
        print("Percent of Rain:{:2f}%".format((countrain/(len(data.weather))*100)))
        print("Percent of Sun:{:2f}%".format((countsun/(len(data.weather))*100)))
        print("Percent of Drizzle:{:2f}%".format((countdrizzle/(len(data.weather))*100)))
        print("Percent of Snow:{:2f}%".format((countsnow/(len(data.weather))*100)))
        print("Percent of Fog:{:2f}%".format((countfog/(len(data.weather))*100)))

        Percent of Rain:43.874059%
        Percent of Sun:43.805613%
        Percent of Drizzle:3.627652%
        Percent of Snow:1.779603%
        Percent of Fog:6.913073%
```

Figure 4:Data Exploration

Exploring dataset, it shows this weather column have 5 types of data i.e., drizzle, rain, sun weather, snow and fog which is counted. whereas, rain and sun weather day have highest count but snow has lowest count. It also shows the percentage value of the weather column’s day types.



Numerical Or Continuous variables

In [9]:
```
data[["precipitation","temp_max","temp_min","wind"]].describe()
```

Out[9]:

	precipitation	temp_max	temp_min	wind
count	1436.000000	1461.000000	1461.000000	1461.000000
mean	3.010237	16.439083	8.234771	3.241136
std	6.654280	7.349758	5.023004	1.437825
min	0.000000	-1.600000	-7.100000	0.400000
25%	0.000000	10.600000	4.400000	2.200000
50%	0.000000	15.600000	8.300000	3.000000
75%	2.800000	22.200000	12.200000	4.000000
max	55.900000	35.600000	18.300000	9.500000

In [10]:
```
data.corr()
```

Out[10]:

	precipitation	temp_max	temp_min	wind
precipitation	1.000000	-0.236824	-0.085276	0.327079
temp_max	-0.236824	1.000000	0.875687	-0.164857
temp_min	-0.085276	0.875687	1.000000	-0.074185
wind	0.327079	-0.164857	-0.074185	1.000000

Figure 5: checking variables type -i

Checking for data types either it is numerical or continuous variables. It shows these are the numerical valued datasets. And find correlation of certain columns in the dataset.

```
In [16]: #using heatmap to visualize
         plt.figure(figsize=(12,7))
         sns.heatmap(data.corr(),annot=True,cmap='coolwarm')

Out[16]: <AxesSubplot:>
```

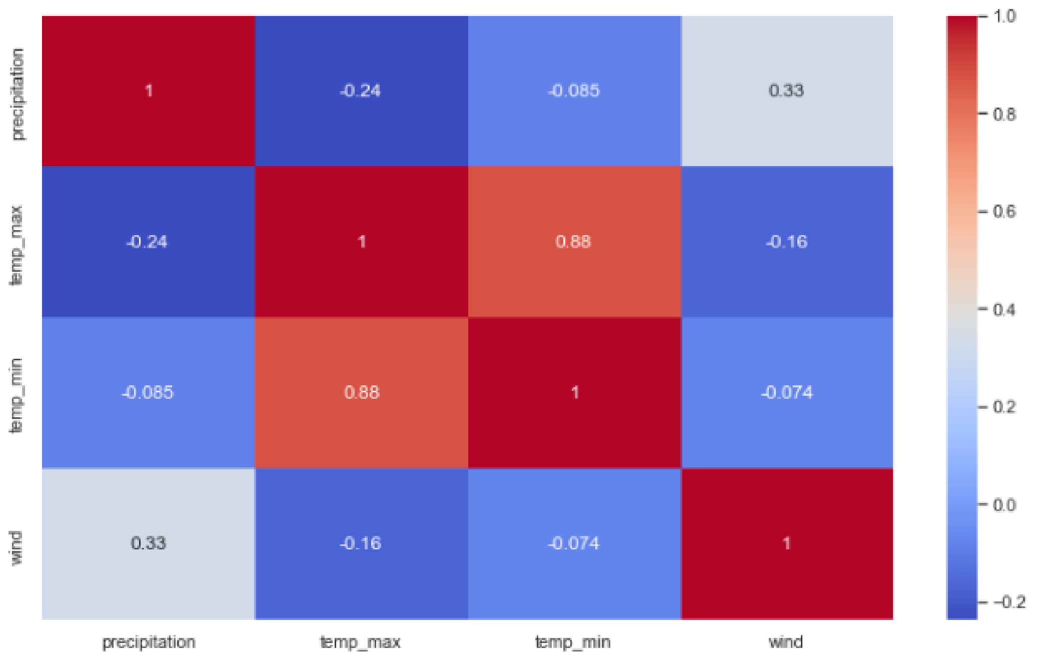
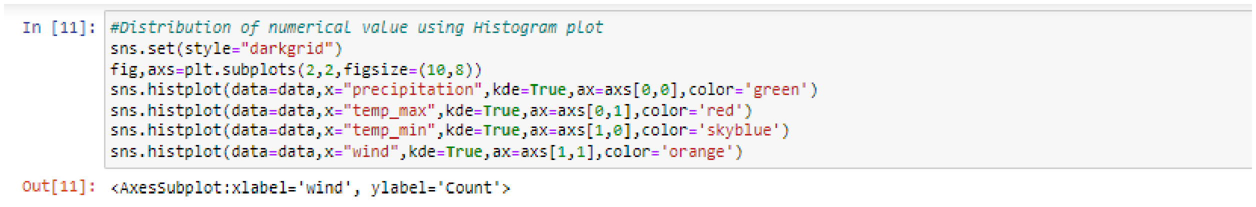
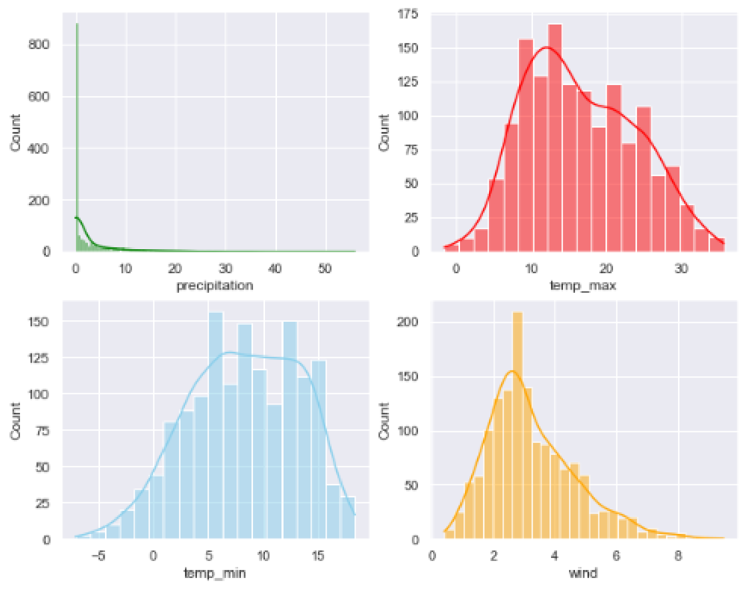


Figure 6: checking variable -ii

Visualizing the correlation of the dataset’s columns using heat map.



```
In [11]: #Distribution of numerical value using Histogram plot
         sns.set(style="darkgrid")
         fig,axs=plt.subplots(2,2,figsize=(10,8))
         sns.histplot(data=data,x="precipitation",kde=True,ax=axs[0,0],color='green')
         sns.histplot(data=data,x="temp_max",kde=True,ax=axs[0,1],color='red')
         sns.histplot(data=data,x="temp_min",kde=True,ax=axs[1,0],color='skyblue')
         sns.histplot(data=data,x="wind",kde=True,ax=axs[1,1],color='orange')

Out[11]: <AxesSubplot:xlabel='wind', ylabel='Count'>
```



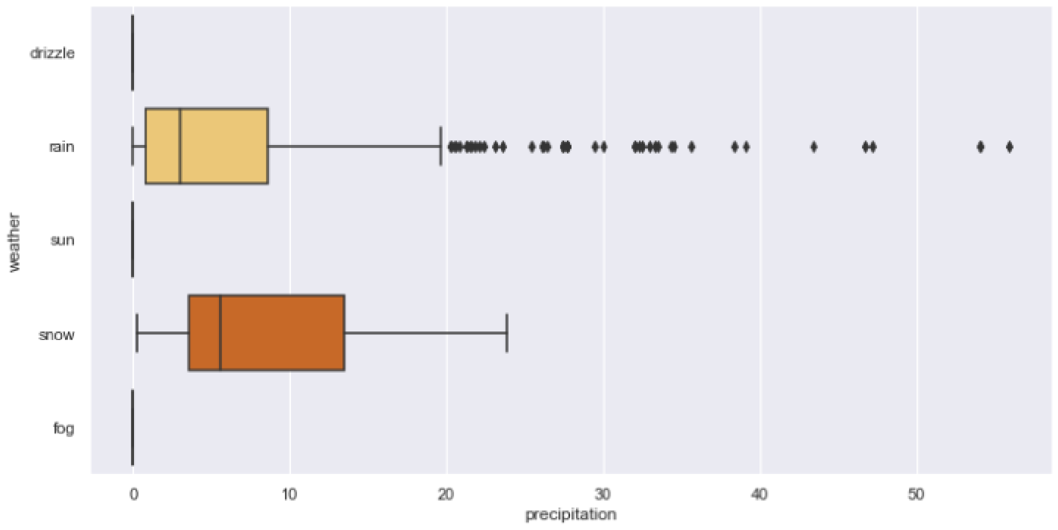
Precipitation and wind are both sensitive to outliers, as shown in the above histogram. And temp min is negatively skewed, with some outliers with both.

Figure 7: checking variables visualization

Displaying above numerical dataset in using histogram plot Outliers affect has both precipitation and wind, as illustrated in the graph image. And the minimum temperature is negatively skewed, with several outliers in both cases.

Finding Outliers

```
In [12]: #plotting data of precipitation and weather columns
         plt.figure(figsize=(12,6))
         sns.boxplot("precipitation","weather",data=data,palette="YlOrBr")

Out[12]: <AxesSubplot:xlabel='precipitation', ylabel='weather'>
```

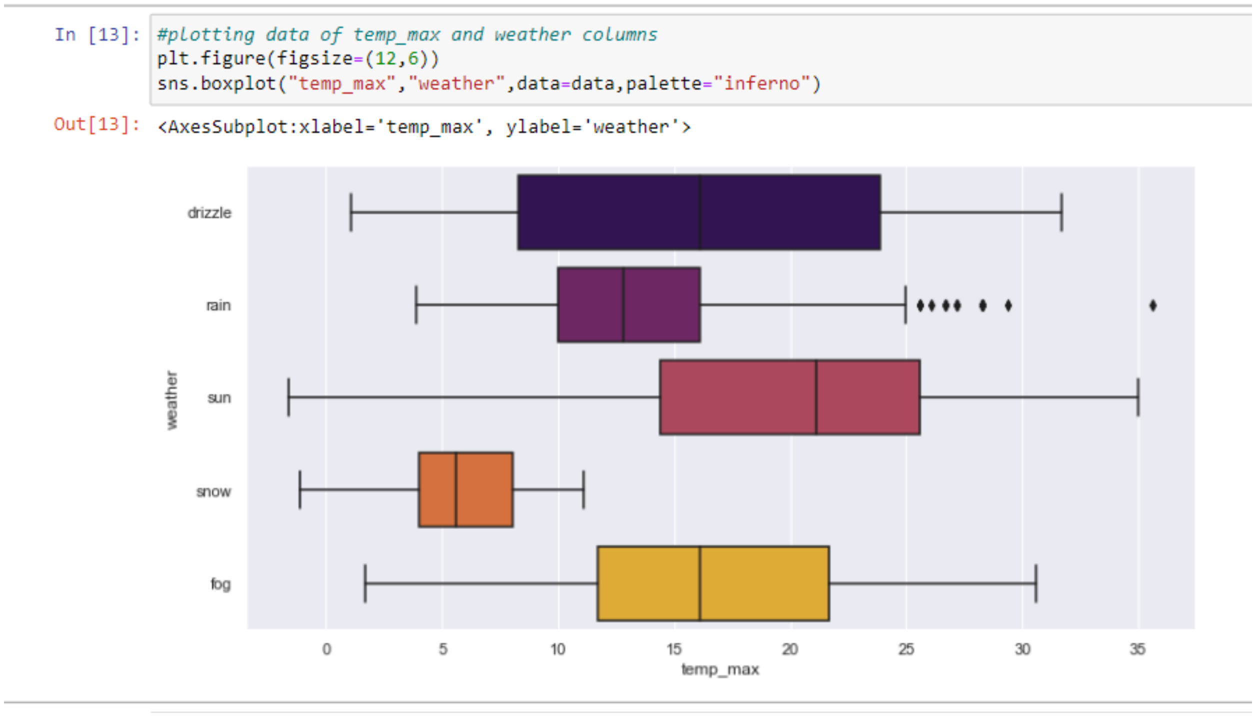


The value comes from the above box plot between Weather and Precipitation.

Rain has a lot of positive outliers, and both Rain and Snow are favorably biased.

Figure 8: finding outlier-i

Finding outlier between precipitation and weather columns of the datasets. It shows, the rain data have many outliers.



```
In [13]: #plotting data of temp_max and weather columns
         plt.figure(figsize=(12,6))
         sns.boxplot("temp_max","weather",data=data,palette="inferno")

Out[13]: <AxesSubplot:xlabel='temp_max', ylabel='weather'>
```

Figure 9: finding Outlier-ii

Finding outliers between temp_max and weather columns. It also shows, the rain has positive outliers.

```
In [14]: #plotting data of wind and weather columns
         plt.figure(figsize=(12,6))
         sns.boxplot("wind","weather",data=data,palette="inferno")

Out[14]: <AxesSubplot:xlabel='wind', ylabel='weather'>
```

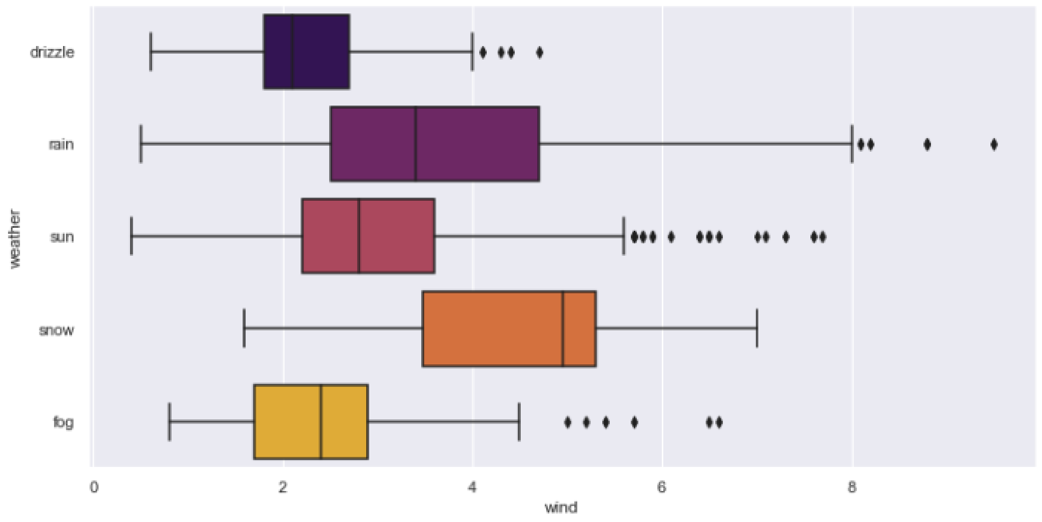
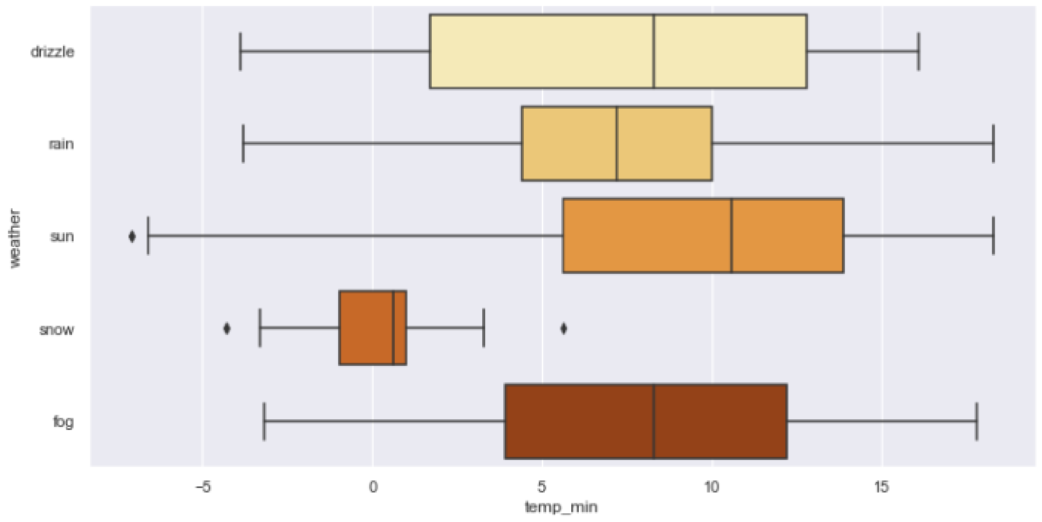


Figure 10: checking outliers-ii

Finding outliers between wind and weather columns. It shows drizzle, rain, sun and fog has many outliers but the snow has none.

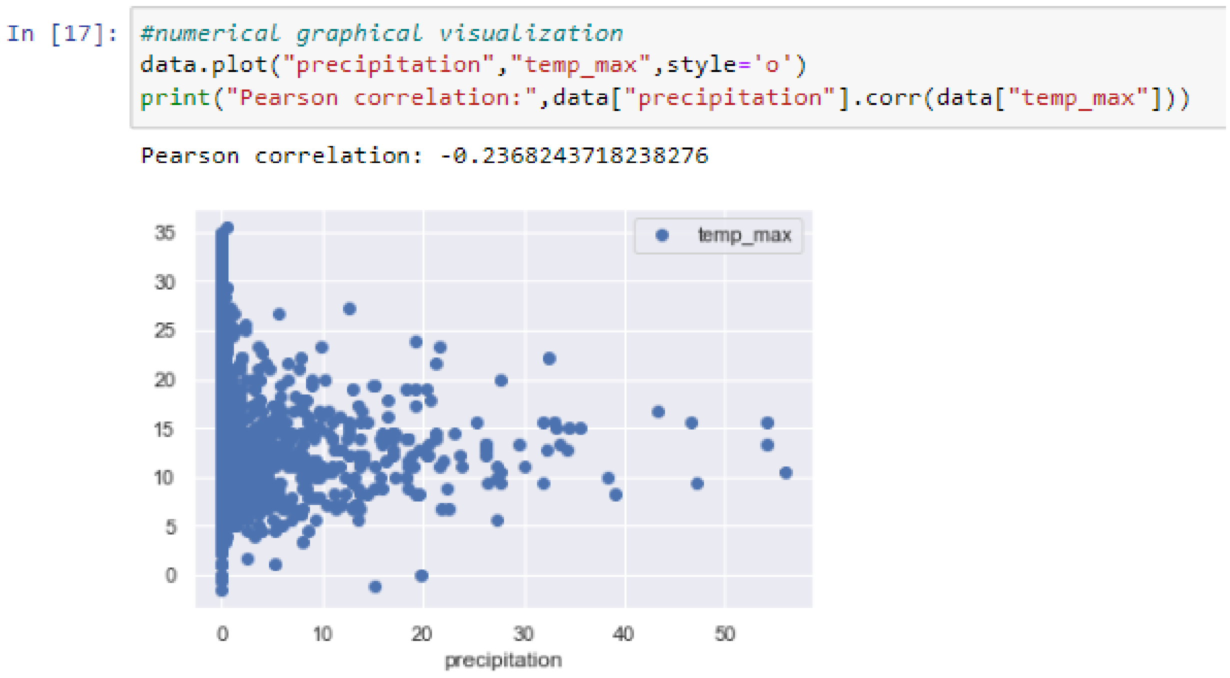
```
In [15]: #plotting data of temp_min and weather columns
         plt.figure(figsize=(12,6))
         sns.boxplot("temp_min","weather",data=data,palette="YlOrBr")

Out[15]: <AxesSubplot:xlabel='temp_min', ylabel='weather'>
```



Some data has negatives outliers, while others have both positively and negatively outliers, and snow is biased negatively.

Figure 11: checking outliers-iii

Finding outliers between temp_min and weather columns. It shows sun have 1 negative outlier and snow have both negative and positive outlier and remaning has none.

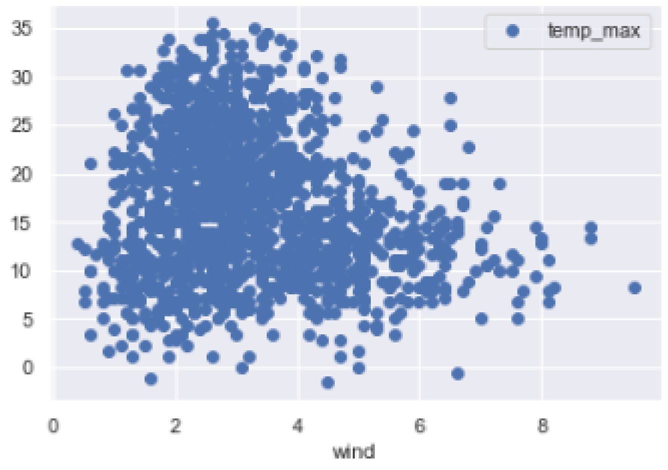


```
In [17]: #numerical graphical visualization
         data.plot("precipitation","temp_max",style='o')
         print("Pearson correlation:",data["precipitation"].corr(data["temp_max"]))

         Pearson correlation: -0.2368243718238276
```

Figure 12: precipitation vs temp_max

Visualization of data between precipitation and temp_max columns.

```
In [18]: data.plot("wind","temp_max",style='o')
         print("Pearson correlation:",data["wind"].corr(data["temp_max"]))

         Pearson correlation: -0.1648566348749548
```



```
In [19]: data.plot("temp_max","temp_min",style='o')

Out[19]: <AxesSubplot:xlabel='temp_max'>
```

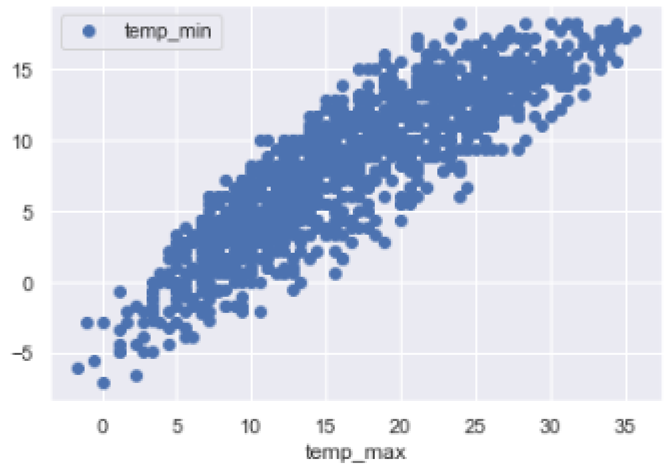
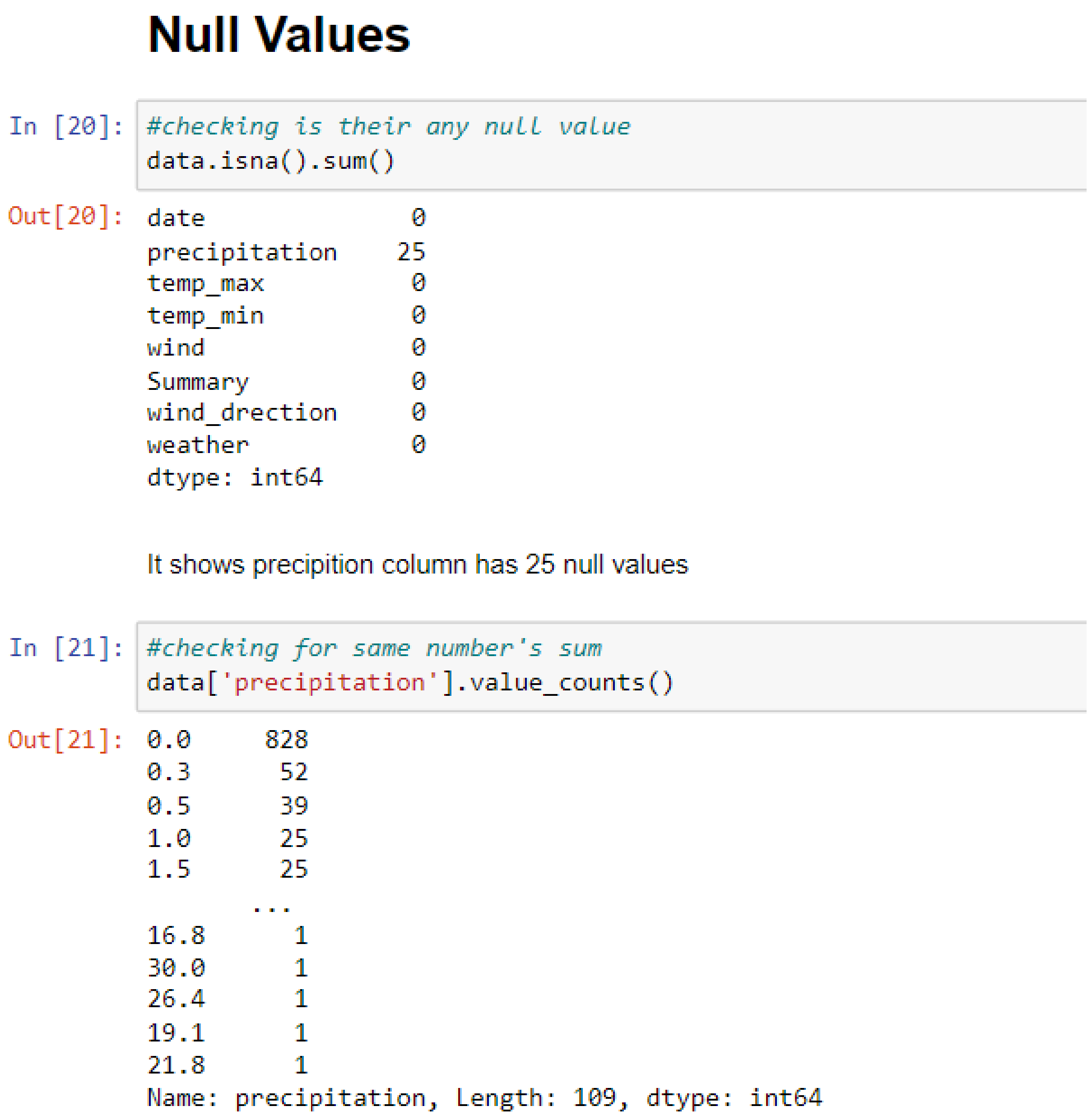


Figure 13: column vs column matplotlib

Visualization of dataset between wind and temp_max and temp_max and temp_min.



Null Values

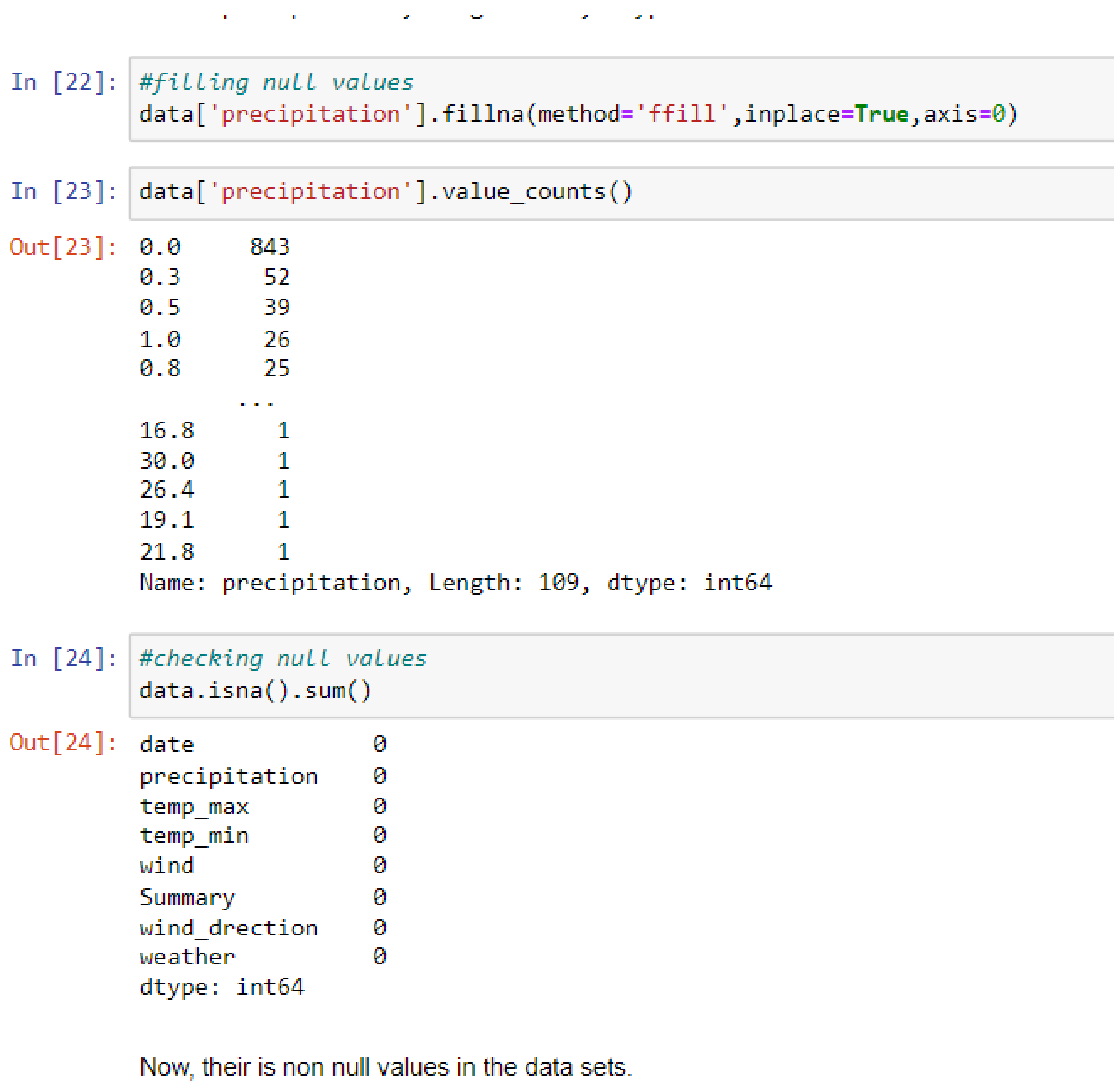
```
In [20]: #checking is their any null value
         data.isna().sum()

Out[20]: date              0
         precipitation    25
         temp_max          0
         temp_min          0
         wind              0
         Summary           0
         wind_drection     0
         weather           0
         dtype: int64
```

It shows precipition column has 25 null values

```
In [21]: #checking for same number's sum
         data['precipitation'].value_counts()

Out[21]: 0.0     828
         0.3      52
         0.5      39
         1.0      25
         1.5      25
                ...
         16.8      1
         30.0      1
         26.4      1
         19.1      1
         21.8      1
         Name: precipitation, Length: 109, dtype: int64
```

Figure 14: Checking null value

Checking for the null values in the datasets. It shows precipitation column have 25 null values. It needs to be clean because cleaning helps to increase the accuracy of the model.

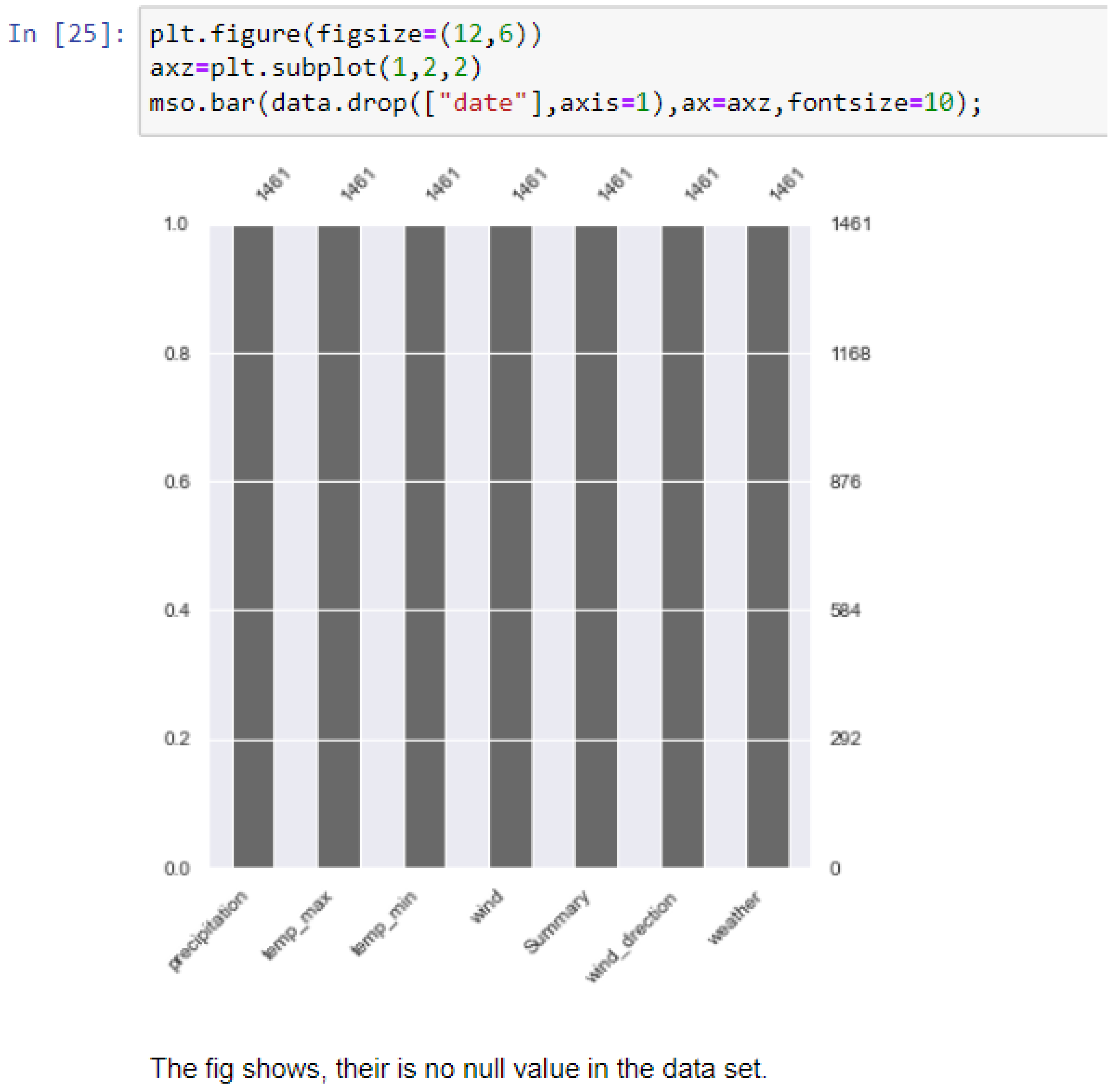


```
In [22]: #filling null values
         data['precipitation'].fillna(method='ffill',inplace=True,axis=0)

In [23]: data['precipitation'].value_counts()

Out[23]: 0.0     843
         0.3      52
         0.5      39
         1.0      26
         0.8      25
                ...
         16.8      1
         30.0      1
         26.4      1
         19.1      1
         21.8      1
         Name: precipitation, Length: 109, dtype: int64

In [24]: #checking null values
         data.isna().sum()

Out[24]: date             0
         precipitation    0
         temp_max         0
         temp_min         0
         wind             0
         Summary          0
         wind_drection    0
         weather          0
         dtype: int64
```

Now, their is non null values in the data sets.

Figure 15: resolving null values

Filling null values with the random average mean value. After filling null value, it shows there is 0 null values.



```
In [25]: plt.figure(figsize=(12,6))
         axz=plt.subplot(1,2,2)
         mso.bar(data.drop(["date"],axis=1),ax=axz,fontsize=10);
```

The fig shows, their is no null value in the data set.

Figure 16: check null value visualization

Using missing library to visualize null values. It shows there is none null values.

Data Preprocessing

```
In [26]: #Droping unnecessary variables becasue it does not affect any changes in our data outcomes.
         df=data.drop(["date"],axis=1)
```

```
In [27]: #Removing the outliers
         Q1=df.quantile(0.25)
         Q3=df.quantile(0.75)
         IQR=Q3-Q1
         df=df[~((df<(Q1-1.5*IQR))|(df>(Q3+1.5*IQR))).any(axis=1)]
```

```
In [28]: #Correction for standard deviation
         df.precipitation=np.sqrt(df.precipitation)
         df.wind=np.sqrt(df.wind)
```

Figure 17: Removing Outliers

Removing the above outlier of the datasets. And dropping the unnecessary column because it does not make any changes to the accuracy of the model.

```
In [29]: #cheking for the outliers
         sns.set(style="darkgrid")
         fig,axs=plt.subplots(2,2,figsize=(10,8))
         sns.histplot(data=df,x="precipitation",kde=True,ax=axs[0,0],color='green')
         sns.histplot(data=df,x="temp_max",kde=True,ax=axs[0,1],color='red')
         sns.histplot(data=df,x="temp_min",kde=True,ax=axs[1,0],color='skyblue')
         sns.histplot(data=df,x="wind",kde=True,ax=axs[1,1],color='orange')

Out[29]: <AxesSubplot:xlabel='wind', ylabel='Count'>
```

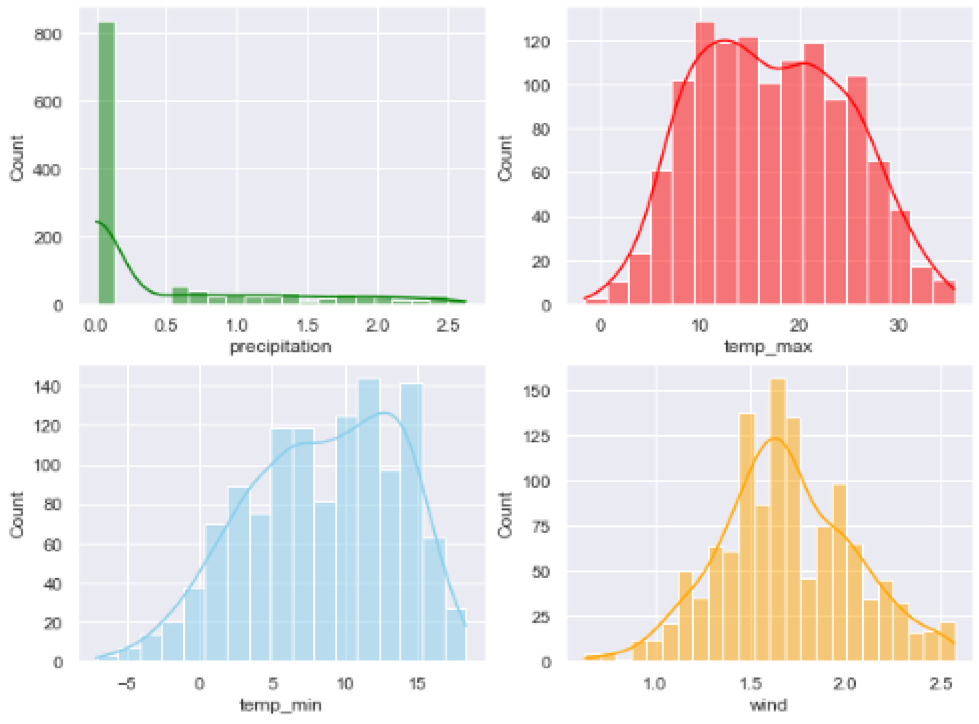
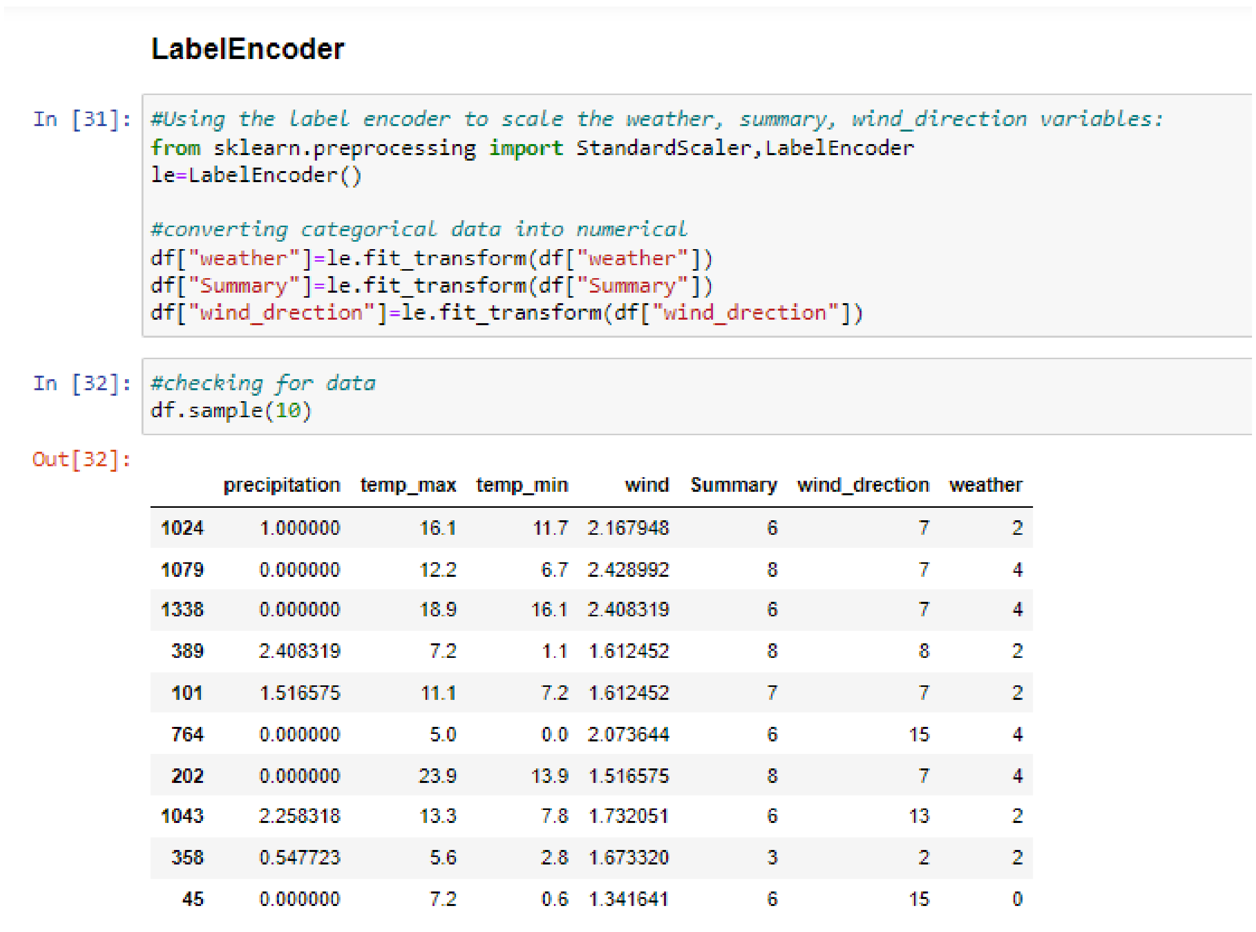


Figure 18: checking outlier's visualizations

After removing outliers, visualizing the dataset again.



LabelEncoder

```
In [31]: #Using the label encoder to scale the weather, summary, wind_direction variables:
         from sklearn.preprocessing import StandardScaler,LabelEncoder
         le=LabelEncoder()

         #converting categorical data into numerical
         df["weather"]=le.fit_transform(df["weather"])
         df["Summary"]=le.fit_transform(df["Summary"])
         df["wind_drection"]=le.fit_transform(df["wind_drection"])
```

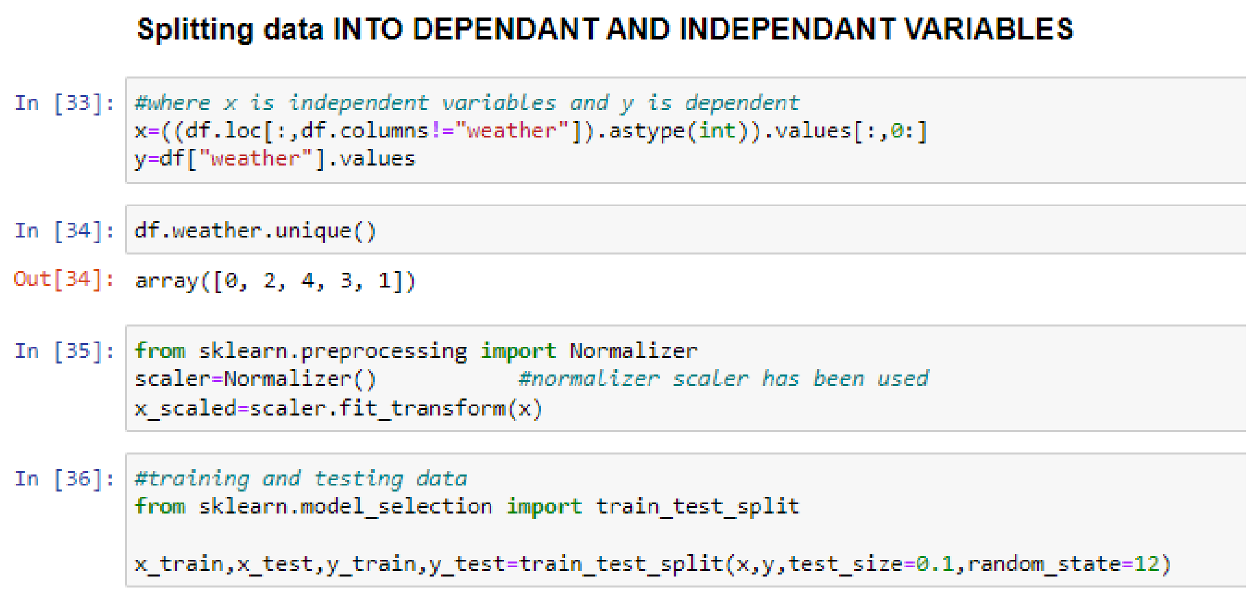
```
In [32]: #checking for data
         df.sample(10)
```

Out[32]:

	precipitation	temp_max	temp_min	wind	Summary	wind_drection	weather
1024	1.000000	16.1	11.7	2.167948	6	7	2
1079	0.000000	12.2	6.7	2.428992	8	7	4
1338	0.000000	18.9	16.1	2.408319	6	7	4
389	2.408319	7.2	1.1	1.612452	8	8	2
101	1.516575	11.1	7.2	1.612452	7	7	2
764	0.000000	5.0	0.0	2.073644	6	15	4
202	0.000000	23.9	13.9	1.516575	8	7	4
1043	2.258318	13.3	7.8	1.732051	6	13	2
358	0.547723	5.6	2.8	1.673320	3	2	2
45	0.000000	7.2	0.6	1.341641	6	15	0

Figure 19: labelEncoding

Converting categorical value into numerical value of the datasets because object values cannot be used for the prediction of the model. Those columns which had object values are converted to numerical value.

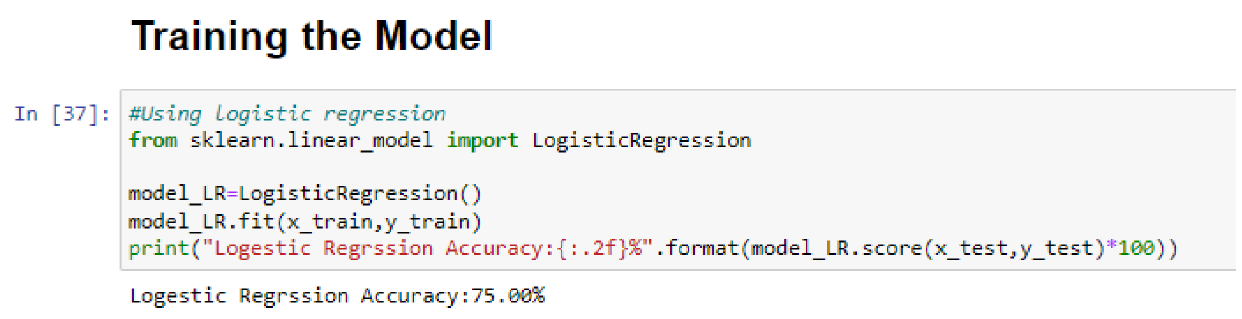


Splitting data INTO DEPENDANT AND INDEPENDANT VARIABLES

```
In [33]: #where x is independent variables and y is dependent
         x=((df.loc[:,df.columns!="weather"]).astype(int)).values[:,0:]
         y=df["weather"].values

In [34]: df.weather.unique()

Out[34]: array([0, 2, 4, 3, 1])

In [35]: from sklearn.preprocessing import Normalizer
         scaler=Normalizer()          #normalizer scaler has been used
         x_scaled=scaler.fit_transform(x)

In [36]: #training and testing data
         from sklearn.model_selection import train_test_split

         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=12)
```

Figure 20: data split

Defining the dependent variables and independents variable. Where, x is an independent variable and y is and dependent variable (target variable). we have split the data as x_train, x_test, y_test, y_train because some data is needed to train and some data needed to test. Whereas, x_train and y_train is used for training datasets and y_test and x_test Is used for testing data sets.

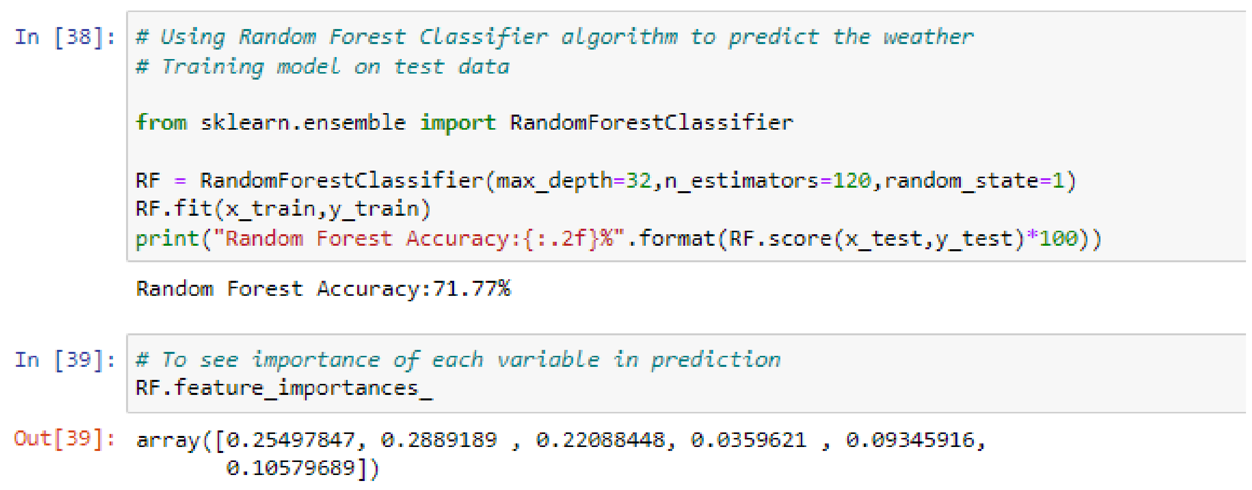


Training the Model

```
In [37]: #Using logistic regression
         from sklearn.linear_model import LogisticRegression

         model_LR=LogisticRegression()
         model_LR.fit(x_train,y_train)
         print("Logestic Regrssion Accuracy:{:.2f}%".format(model_LR.score(x_test,y_test)*100))

         Logestic Regrssion Accuracy:75.00%
```

Figure 21: logistic regrssion training

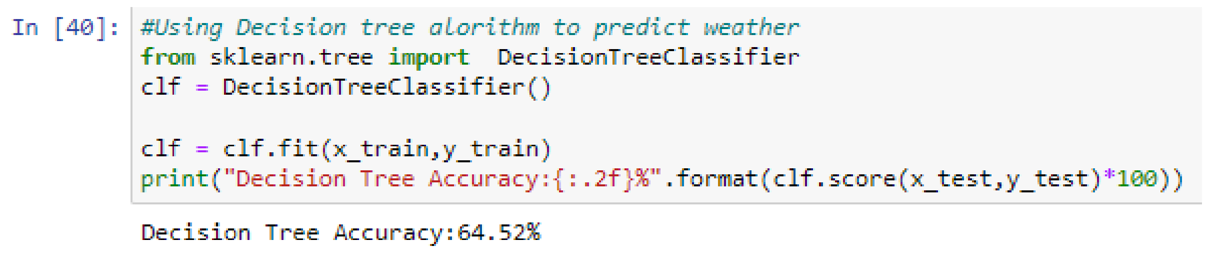
Using logistic regression to train the model. The following model displays the 75% accuracy of the prediction.



```
In [38]: # Using Random Forest Classifier algorithm to predict the weather
         # Training model on test data

         from sklearn.ensemble import RandomForestClassifier

         RF = RandomForestClassifier(max_depth=32,n_estimators=120,random_state=1)
         RF.fit(x_train,y_train)
         print("Random Forest Accuracy:{:.2f}%".format(RF.score(x_test,y_test)*100))

         Random Forest Accuracy:71.77%

In [39]: # To see importance of each variable in prediction
         RF.feature_importances_

Out[39]: array([0.25497847, 0.2889189 , 0.22088448, 0.0359621 , 0.09345916,
                0.10579689])
```

Figure 22: Random Forest model

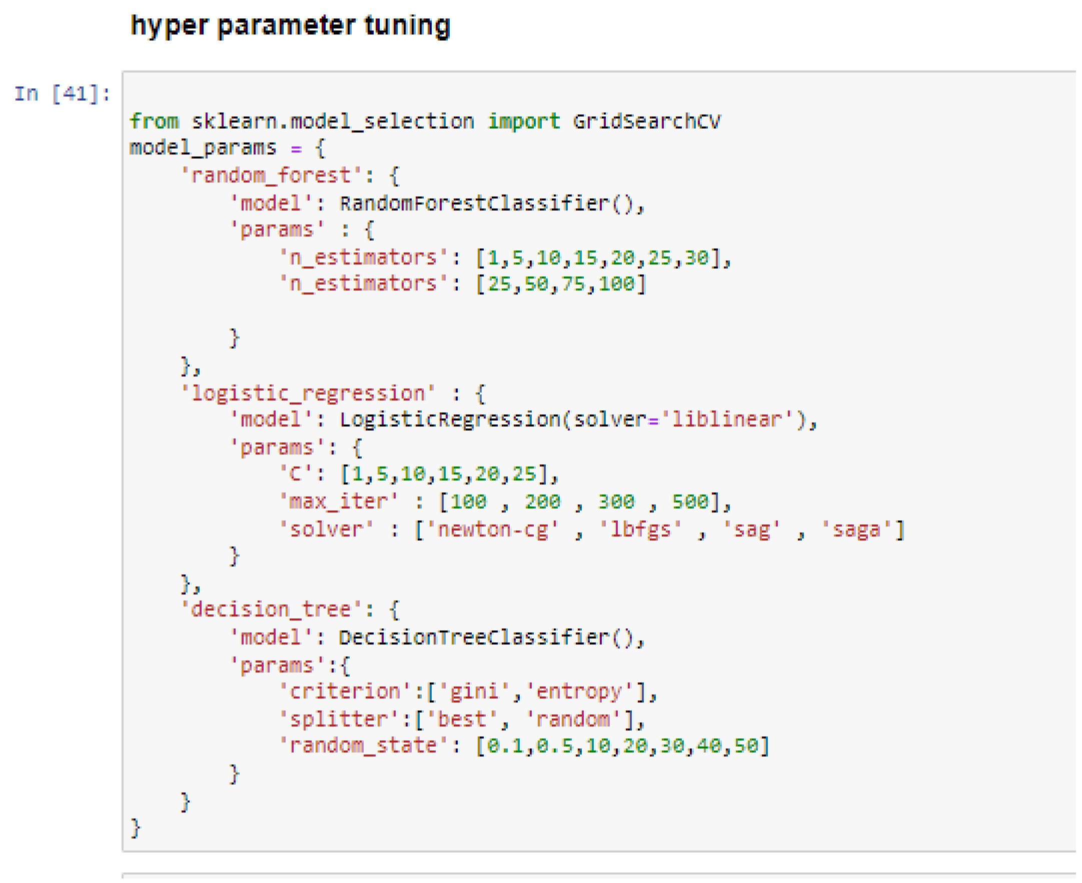
Using Random Forest model training to find the accuracy. It shows 71.77% accuracy of the prediction which is less accuracy than logistic regression.



```
In [40]: #Using Decision tree alorithm to predict weather
         from sklearn.tree import  DecisionTreeClassifier
         clf = DecisionTreeClassifier()

         clf = clf.fit(x_train,y_train)
         print("Decision Tree Accuracy:{:.2f}%".format(clf.score(x_test,y_test)*100))

         Decision Tree Accuracy:64.52%
```

Figure 23: Decision Tree algorithm

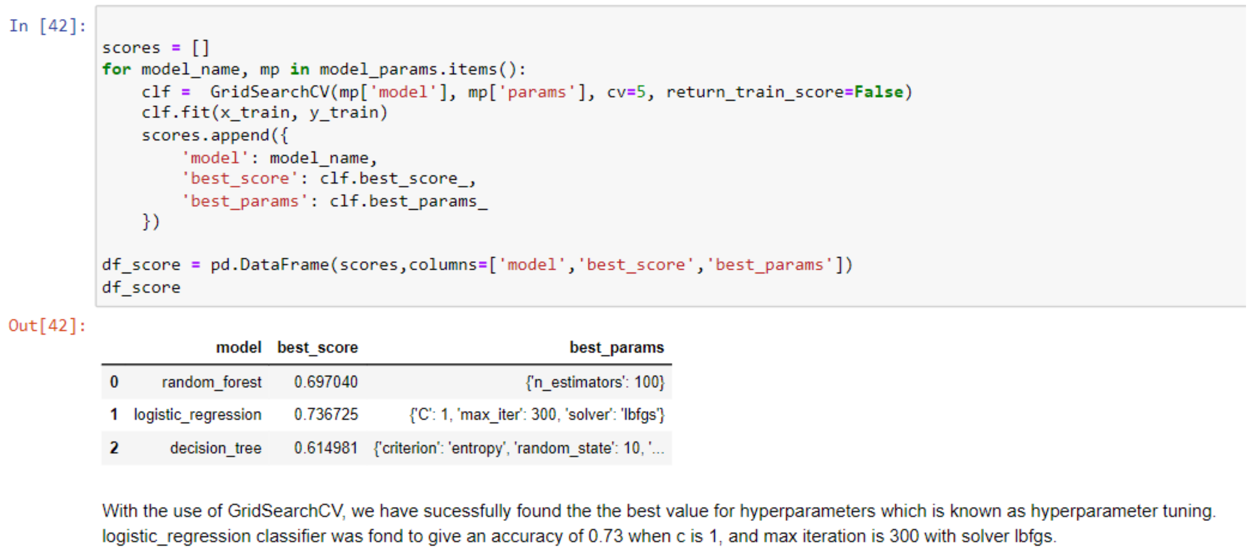
Using Decision Tree to find accuracy. It shows 64.52% accuracy of the prediction which is lesser than logistic and random forest model.


hyper parameter tuning

In [41]:
```
from sklearn.model_selection import GridSearchCV
model_params = {
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'n_estimators': [1,5,10,15,20,25,30],
            'n_estimators': [25,50,75,100]

        }
    },
    'logistic_regression' : {
        'model': LogisticRegression(solver='liblinear'),
        'params': {
            'C': [1,5,10,15,20,25],
            'max_iter' : [100 , 200 , 300 , 500],
            'solver' : ['newton-cg' , 'lbfgs' , 'sag' , 'saga']
        }
    },
    'decision_tree': {
        'model': DecisionTreeClassifier(),
        'params':{
            'criterion':['gini','entropy'],
            'splitter':['best', 'random'],
            'random_state': [0.1,0.5,10,20,30,40,50]
        }
    }
}
```

Figure 24: hyper parameter tuning

Using Grid Search CV to process Hyper parameter tuning the above models i.e., random forest, logstic_regression and decision tree. It helps to know the best hyper parameter to get best accuracy of the model. Each model has hyper parameter and various values



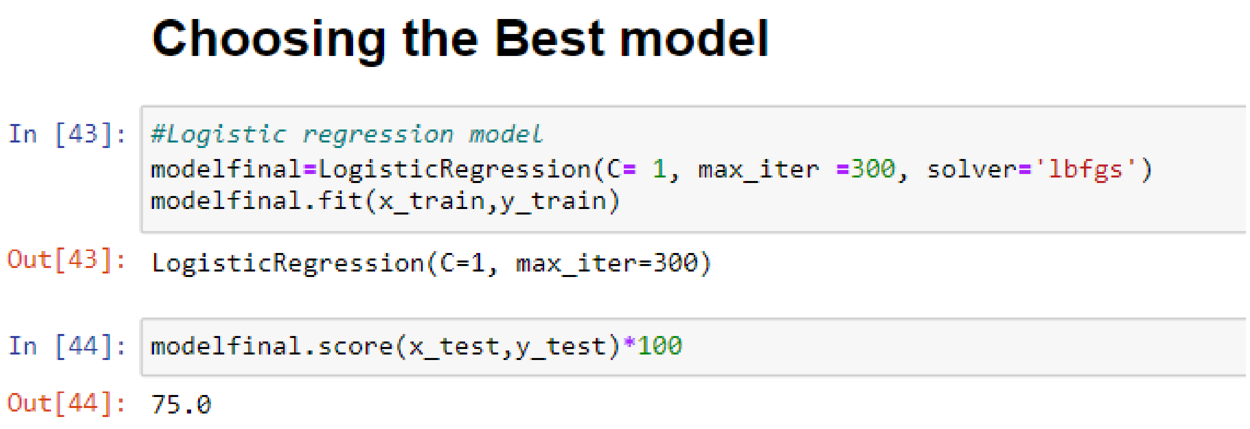
```
In [42]:
scores = []
for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=5, return_train_score=False)
    clf.fit(x_train, y_train)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

df_score = pd.DataFrame(scores,columns=['model','best_score','best_params'])
df_score
Out[42]:
```

	model	best_score	best_params
0	random_forest	0.697040	{'n_estimators': 100}
1	logistic_regression	0.736725	{'C': 1, 'max_iter': 300, 'solver': 'lbfgs'}
2	decision_tree	0.614981	{'criterion': 'entropy', 'random_state': 10, '...

With the use of GridSearchCV, we have sucessfully found the the best value for hyperparameters which is known as hyperparameter tuning. logistic_regression classifier was fond to give an accuracy of 0.73 when c is 1, and max iteration is 300 with solver lbfgs.

Figure 25: Best Parameter

Outputting the best model and best parameter. The following output shows logistic_regression has best score (0.73) and its best parameters are “C” is 1, “max_iter” is 300 and “solver” is lbfgs. And the worst model is decision tree with lowest score (0.61) with its best parameters.



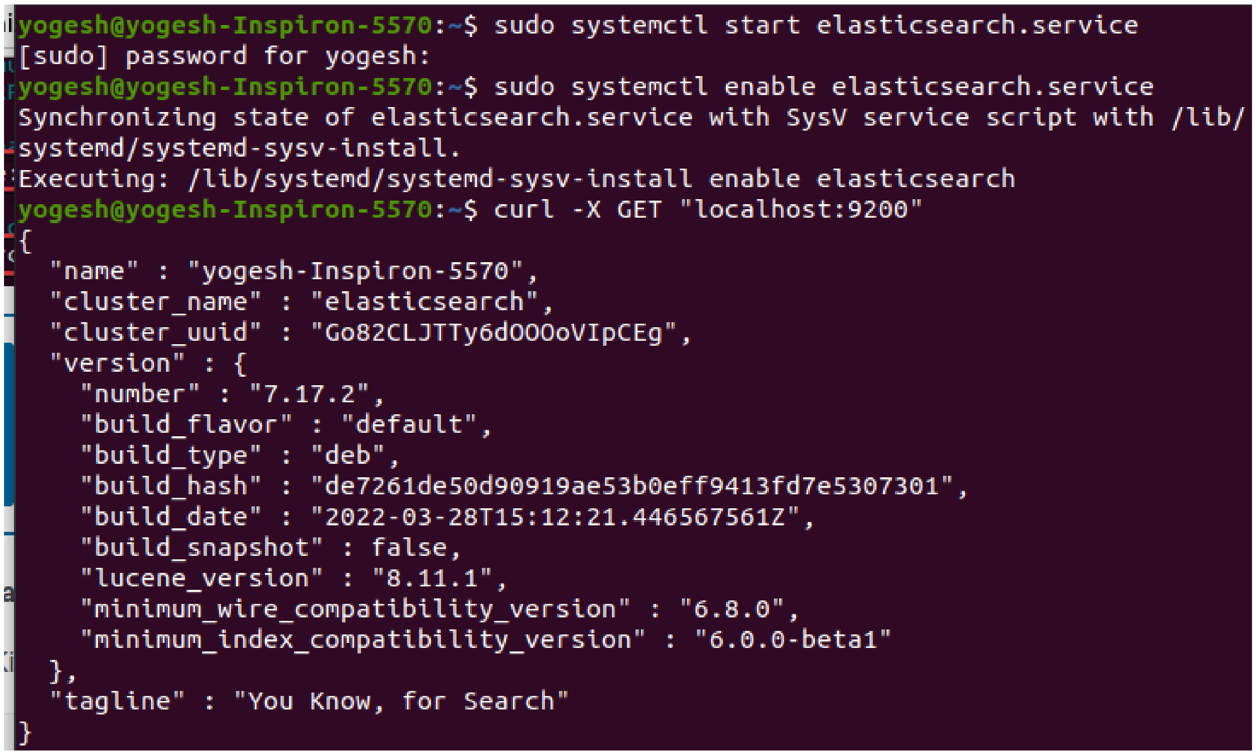
Choosing the Best model

```
In [43]: #Logistic regression model
         modelfinal=LogisticRegression(C= 1, max_iter =300, solver='lbfgs')
         modelfinal.fit(x_train,y_train)
Out[43]: LogisticRegression(C=1, max_iter=300)

In [44]: modelfinal.score(x_test,y_test)*100
Out[44]: 75.0
```

Figure 26: choosing the best model

Choosing the best model which is Logistic Regression and best hyper parameters of its. The final model of this model shows 75.0 score.

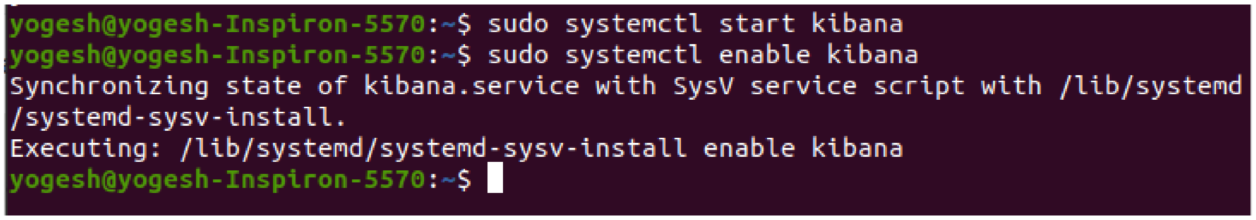
Using Elastic Search



```
yogesh@yogesh-Inspiron-5570:~$ sudo systemctl start elasticsearch.service
[sudo] password for yogesh:
yogesh@yogesh-Inspiron-5570:~$ sudo systemctl enable elasticsearch.service
Synchronizing state of elasticsearch.service with SysV service script with /lib/
systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable elasticsearch
yogesh@yogesh-Inspiron-5570:~$ curl -X GET "localhost:9200"
{
  "name" : "yogesh-Inspiron-5570",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "Go82CLJTTy6dOOOoVIpCEg",
  "version" : {
    "number" : "7.17.2",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "de7261de50d90919ae53b0eff9413fd7e5307301",
    "build_date" : "2022-03-28T15:12:21.446567561Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Starting elastic search for analytical purpose.

Figure 27: starting elastic search



```
yogesh@yogesh-Inspiron-5570:~$ sudo systemctl start kibana
yogesh@yogesh-Inspiron-5570:~$ sudo systemctl enable kibana
Synchronizing state of kibana.service with SysV service script with /lib/systemd
/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable kibana
yogesh@yogesh-Inspiron-5570:~$ 
```

Figure 28: starting Kibana

Running Kibana to visualize the data.

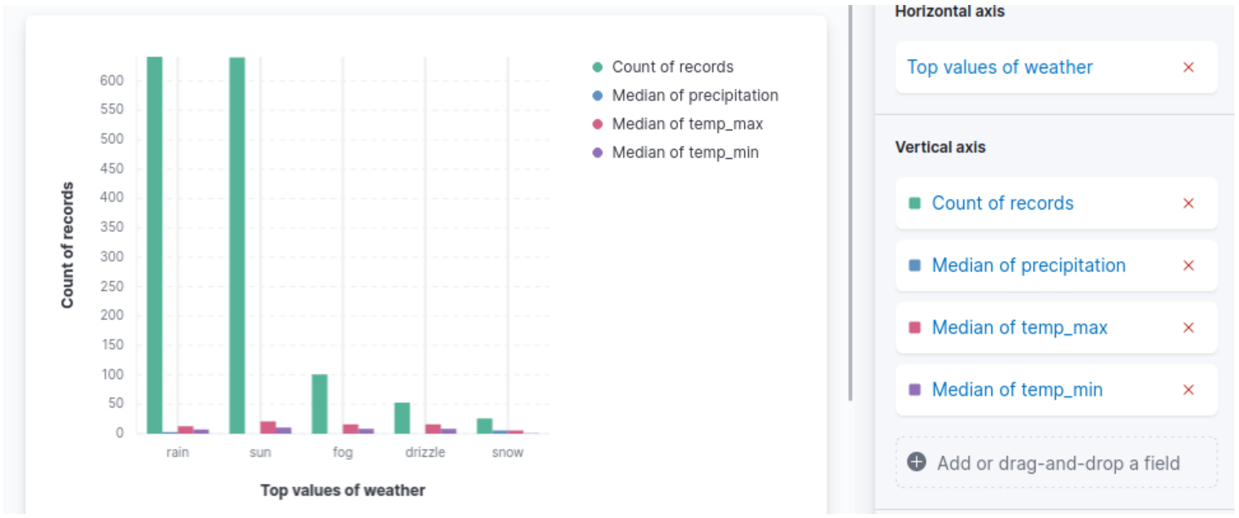


Figure 29: Elasticsearch visualization -i

The following visualization shows the count of weather data. Green bar represents to count of records, blue bar represents the median of precipitation, red bar shows the temp_max and purple bar shows the median of temp_min.

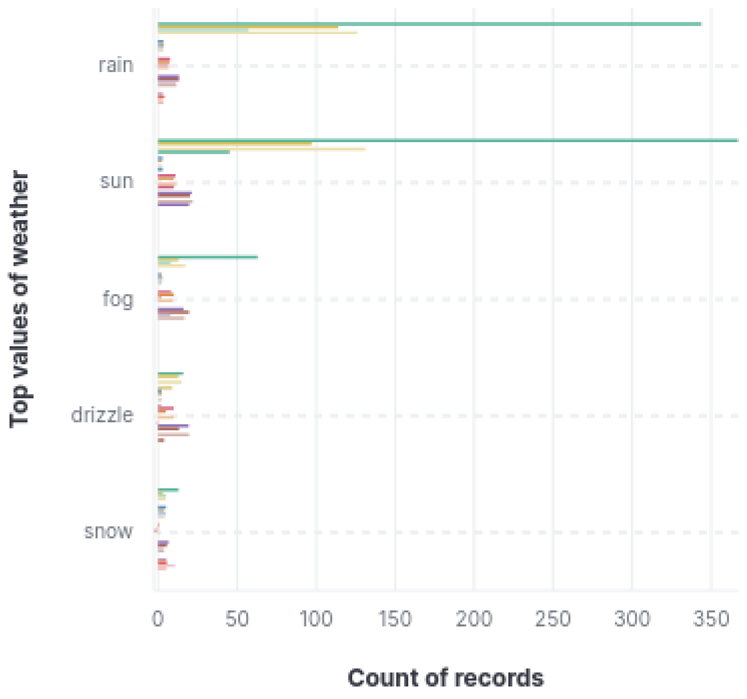
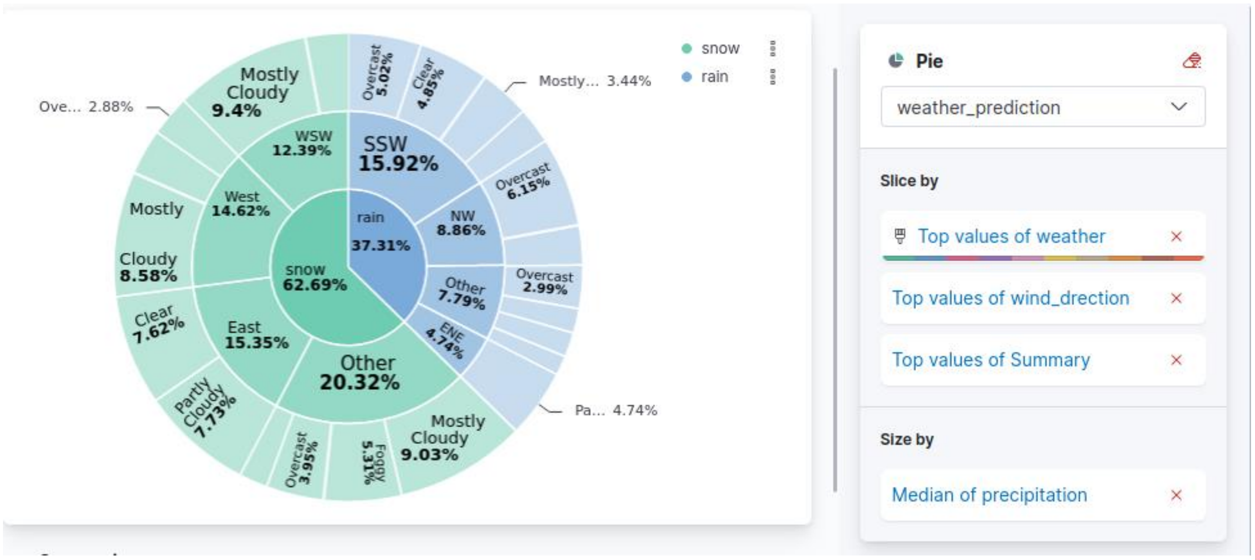


Figure 30: Elasticsearch visualization -ii

This figure shows the weathers data and other independent variables.



Pie

weather_prediction

Slice by

Top values of weather

Top values of wind_drection

Top values of Summary

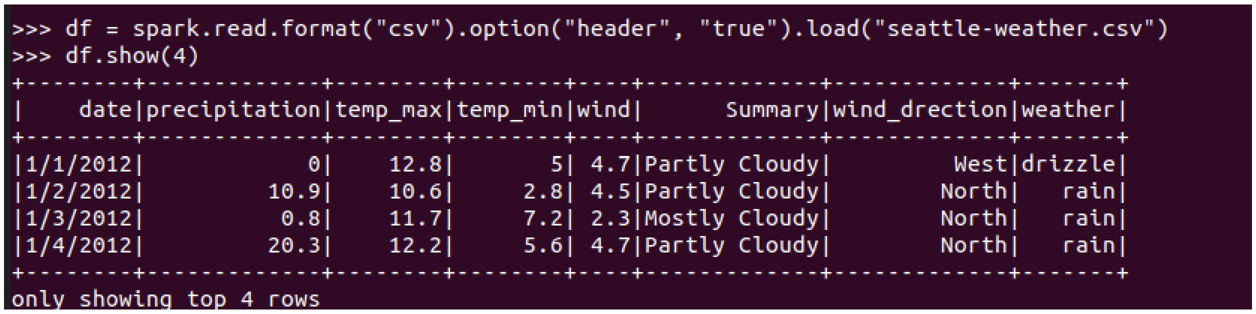
Size by

Median of precipitation

Figure 31: Elasticsearch -iii

The pie-chart shows the data of weather, wind_direction, and summary sized by precipitation.

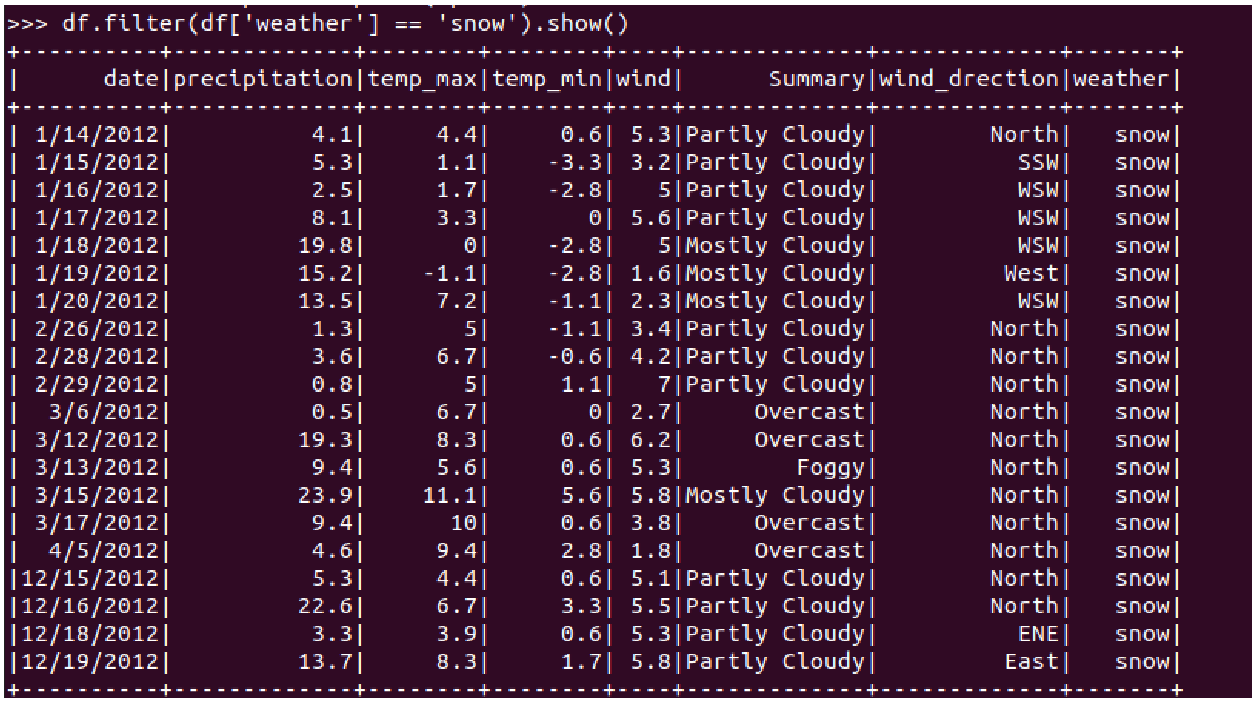
Using Spark



```
>>> df = spark.read.format("csv").option("header", "true").load("seattle-weather.csv")
>>> df.show(4)
+--------+-------------+--------+--------+----+-------------+-------------+-------+
|    date|precipitation|temp_max|temp_min|wind|      Summary|wind_drection|weather|
+--------+-------------+--------+--------+----+-------------+-------------+-------+
|1/1/2012|            0|    12.8|       5| 4.7|Partly Cloudy|         West|drizzle|
|1/2/2012|         10.9|    10.6|     2.8| 4.5|Partly Cloudy|        North|   rain|
|1/3/2012|          0.8|    11.7|     7.2| 2.3|Mostly Cloudy|        North|   rain|
|1/4/2012|         20.3|    12.2|     5.6| 4.7|Partly Cloudy|        North|   rain|
+--------+-------------+--------+--------+----+-------------+-------------+-------+
only showing top 4 rows
```

Figure 32: spark csv reading

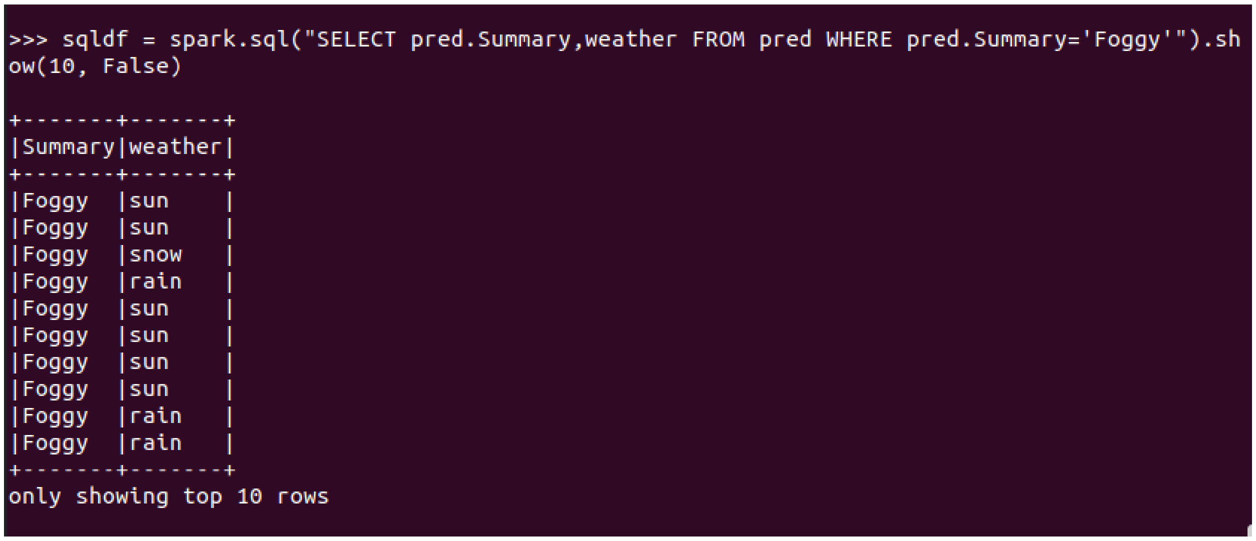
Reading csv file in spark.



```
>>> df.filter(df['weather'] == 'snow').show()
+----------+-------------+--------+--------+----+-------------+------------+-------+
|      date|precipitation|temp_max|temp_min|wind|      Summary|wind_drection|weather|
+----------+-------------+--------+--------+----+-------------+------------+-------+
| 1/14/2012|          4.1|     4.4|     0.6| 5.3|Partly Cloudy|       North|   snow|
| 1/15/2012|          5.3|     1.1|    -3.3| 3.2|Partly Cloudy|         SSW|   snow|
| 1/16/2012|          2.5|     1.7|    -2.8|   5|Partly Cloudy|         WSW|   snow|
| 1/17/2012|          8.1|     3.3|       0| 5.6|Partly Cloudy|         WSW|   snow|
| 1/18/2012|         19.8|       0|    -2.8|   5|Mostly Cloudy|         WSW|   snow|
| 1/19/2012|         15.2|    -1.1|    -2.8| 1.6|Mostly Cloudy|        West|   snow|
| 1/20/2012|         13.5|     7.2|    -1.1| 2.3|Mostly Cloudy|         WSW|   snow|
| 2/26/2012|          1.3|       5|    -1.1| 3.4|Partly Cloudy|       North|   snow|
| 2/28/2012|          3.6|     6.7|    -0.6| 4.2|Partly Cloudy|       North|   snow|
| 2/29/2012|          0.8|       5|     1.1|   7|Partly Cloudy|       North|   snow|
|  3/6/2012|          0.5|     6.7|       0| 2.7|     Overcast|       North|   snow|
| 3/12/2012|         19.3|     8.3|     0.6| 6.2|     Overcast|       North|   snow|
| 3/13/2012|          9.4|     5.6|     0.6| 5.3|        Foggy|       North|   snow|
| 3/15/2012|         23.9|    11.1|     5.6| 5.8|Mostly Cloudy|       North|   snow|
| 3/17/2012|          9.4|      10|     0.6| 3.8|     Overcast|       North|   snow|
|  4/5/2012|          4.6|     9.4|     2.8| 1.8|     Overcast|       North|   snow|
|12/15/2012|          5.3|     4.4|     0.6| 5.1|Partly Cloudy|       North|   snow|
|12/16/2012|         22.6|     6.7|     3.3| 5.5|Partly Cloudy|       North|   snow|
|12/18/2012|          3.3|     3.9|     0.6| 5.3|Partly Cloudy|         ENE|   snow|
|12/19/2012|         13.7|     8.3|     1.7| 5.8|Partly Cloudy|        East|   snow|
+----------+-------------+--------+--------+----+-------------+------------+-------+
```

Figure 33: spark SQL queries

Filter find those data whose weather column have snow value.



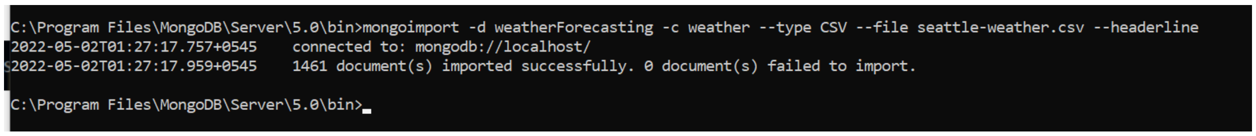
```
>>> sqldf = spark.sql("SELECT pred.Summary,weather FROM pred WHERE pred.Summary='Foggy'").sh
ow(10, False)

+-------+-------+
|Summary|weather|
+-------+-------+
|Foggy  |sun    |
|Foggy  |sun    |
|Foggy  |snow   |
|Foggy  |rain   |
|Foggy  |sun    |
|Foggy  |sun    |
|Foggy  |sun    |
|Foggy  |sun    |
|Foggy  |rain   |
|Foggy  |rain   |
+-------+-------+
only showing top 10 rows
```

Figure 34: Spark sql queries

Find those data which have foggy and value which is limited to 10.

Using MongoDB

Firstly, data can be imported in mongoDB environment by using the some sort of command line as given below.

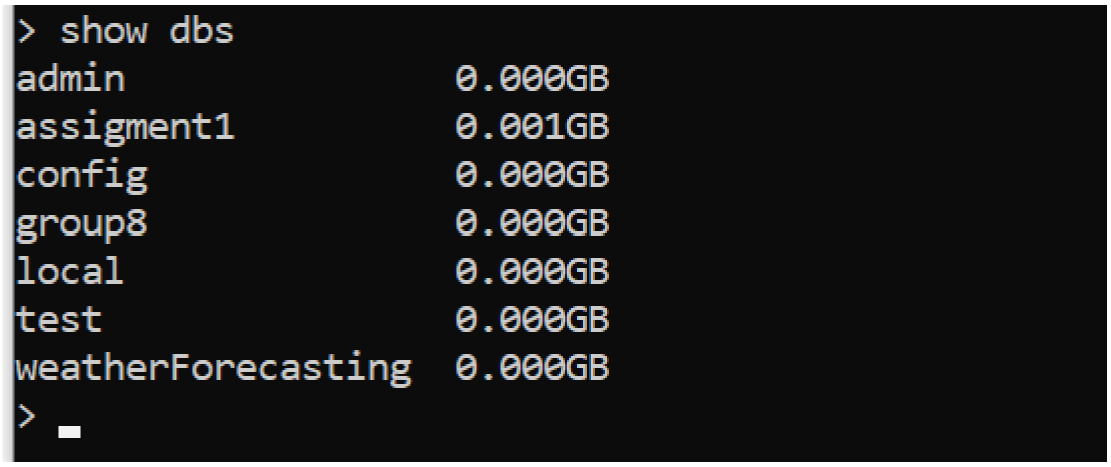


```
C:\Program Files\MongoDB\Server\5.0\bin>mongoimport -d weatherForecasting -c weather --type CSV --file seattle-weather.csv --headerline
2022-05-02T01:27:17.757+0545    connected to: mongodb://localhost/
2022-05-02T01:27:17.959+0545    1461 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\5.0\bin>
```

Figure 35 Import CSV File

In the above command line "mongoimport" is used to import the any type of dataset in mongoDB environment. "-d" stand for database where, database name is weatherForcasting, "-c" stand for collection which is equivalent to relation table and collection name is weather with the type of CSV file. The "-headerline" generally represent the list of the outlier in the dataset. (habilelabs, 2022)

show database

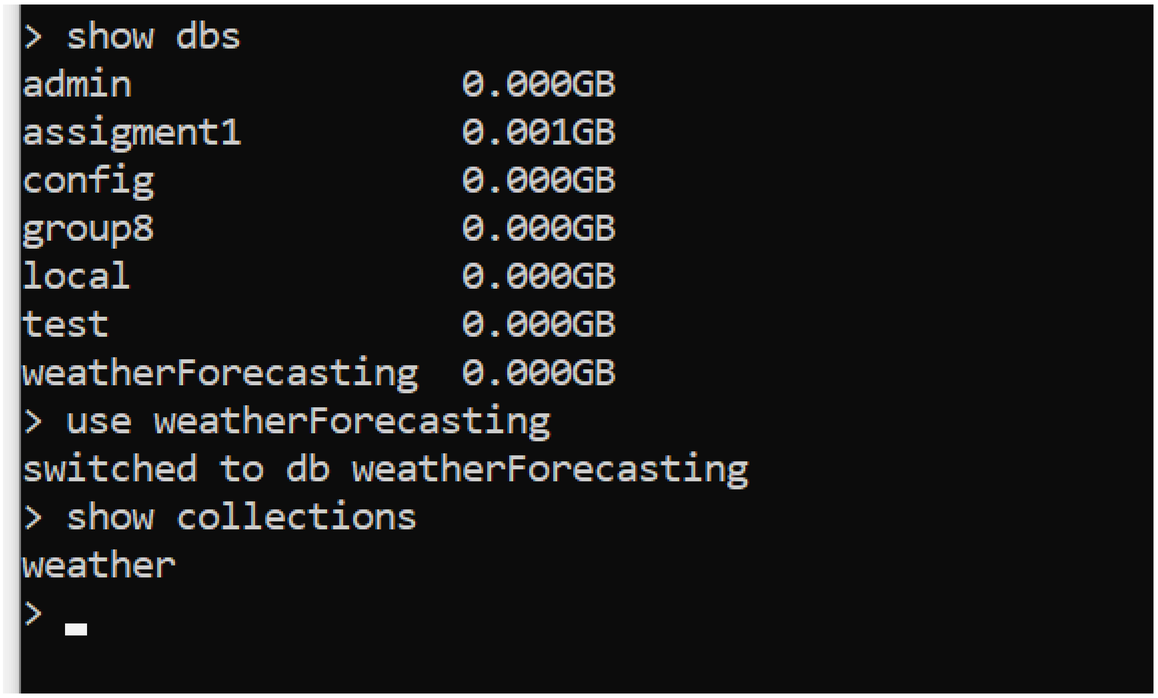


```
> show dbs
admin               0.000GB
assigment1          0.001GB
config              0.000GB
group8              0.000GB
local               0.000GB
test                0.000GB
weatherForecasting  0.000GB
>
```

Figure 36 Show Databases

To show the database, show dbs command is used to total number of database available in the mongoDB environment, in the above figure there are seven databases but this report only "weatherForecasting" database is in used to retrieve datasets.

Switch to weatherForecasting Database

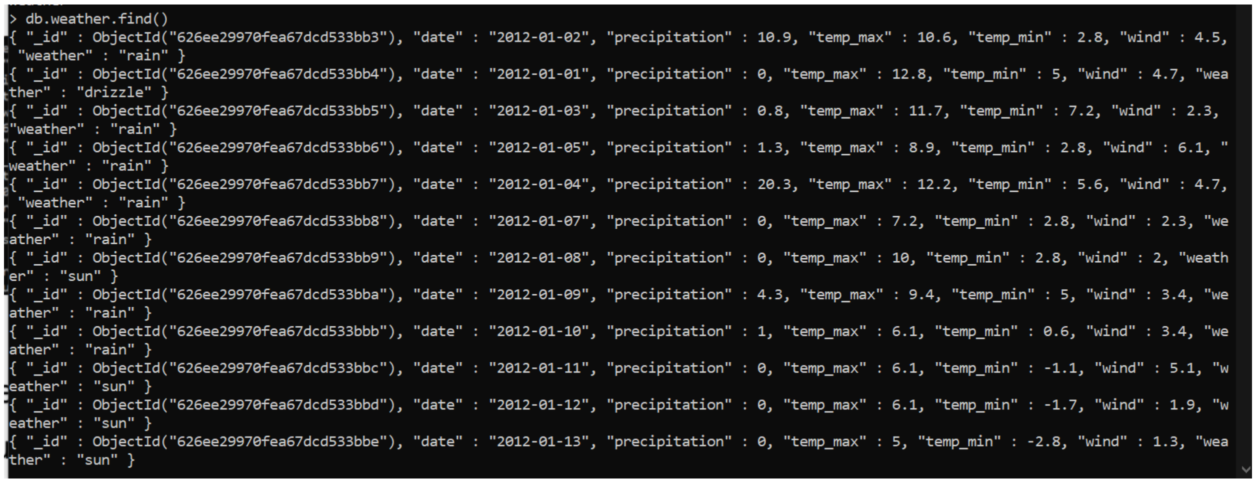


```
> show dbs
admin               0.000GB
assigment1          0.001GB
config              0.000GB
group8              0.000GB
local               0.000GB
test                0.000GB
weatherForecasting  0.000GB
> use weatherForecasting
switched to db weatherForecasting
> show collections
weather
>
```

Figure 37 Switch into database

By using the use command, switch into the weatherForcasting database which contains the weather collection that mean table.

Show collection dataset

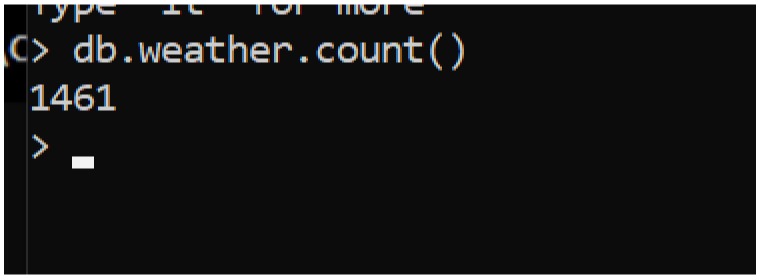


```
> db.weather.find()
{ "_id" : ObjectId("626ee29970fea67dcd533bb3"), "date" : "2012-01-02", "precipitation" : 10.9, "temp_max" : 10.6, "temp_min" : 2.8, "wind" : 4.5,
 "weather" : "rain" }
{ "_id" : ObjectId("626ee29970fea67dcd533bb4"), "date" : "2012-01-01", "precipitation" : 0, "temp_max" : 12.8, "temp_min" : 5, "wind" : 4.7, "wea
ther" : "drizzle" }
{ "_id" : ObjectId("626ee29970fea67dcd533bb5"), "date" : "2012-01-03", "precipitation" : 0.8, "temp_max" : 11.7, "temp_min" : 7.2, "wind" : 2.3,
"weather" : "rain" }
{ "_id" : ObjectId("626ee29970fea67dcd533bb6"), "date" : "2012-01-05", "precipitation" : 1.3, "temp_max" : 8.9, "temp_min" : 2.8, "wind" : 6.1, "
weather" : "rain" }
{ "_id" : ObjectId("626ee29970fea67dcd533bb7"), "date" : "2012-01-04", "precipitation" : 20.3, "temp_max" : 12.2, "temp_min" : 5.6, "wind" : 4.7,
 "weather" : "rain" }
{ "_id" : ObjectId("626ee29970fea67dcd533bb8"), "date" : "2012-01-07", "precipitation" : 0, "temp_max" : 7.2, "temp_min" : 2.8, "wind" : 2.3, "we
ather" : "rain" }
{ "_id" : ObjectId("626ee29970fea67dcd533bb9"), "date" : "2012-01-08", "precipitation" : 0, "temp_max" : 10, "temp_min" : 2.8, "wind" : 2, "weath
er" : "sun" }
{ "_id" : ObjectId("626ee29970fea67dcd533bba"), "date" : "2012-01-09", "precipitation" : 4.3, "temp_max" : 9.4, "temp_min" : 5, "wind" : 3.4, "we
ather" : "rain" }
{ "_id" : ObjectId("626ee29970fea67dcd533bbb"), "date" : "2012-01-10", "precipitation" : 1, "temp_max" : 6.1, "temp_min" : 0.6, "wind" : 3.4, "we
ather" : "rain" }
{ "_id" : ObjectId("626ee29970fea67dcd533bbc"), "date" : "2012-01-11", "precipitation" : 0, "temp_max" : 6.1, "temp_min" : -1.1, "wind" : 5.1, "w
eather" : "sun" }
{ "_id" : ObjectId("626ee29970fea67dcd533bbd"), "date" : "2012-01-12", "precipitation" : 0, "temp_max" : 6.1, "temp_min" : -1.7, "wind" : 1.9, "w
eather" : "sun" }
{ "_id" : ObjectId("626ee29970fea67dcd533bbe"), "date" : "2012-01-13", "precipitation" : 0, "temp_max" : 5, "temp_min" : -2.8, "wind" : 1.3, "wea
ther" : "sun" }
```

Figure 38 Show collection data

By using “db.weather.find()” command, all dataset can be displayed as shown in the above figure.

Count number of Documents

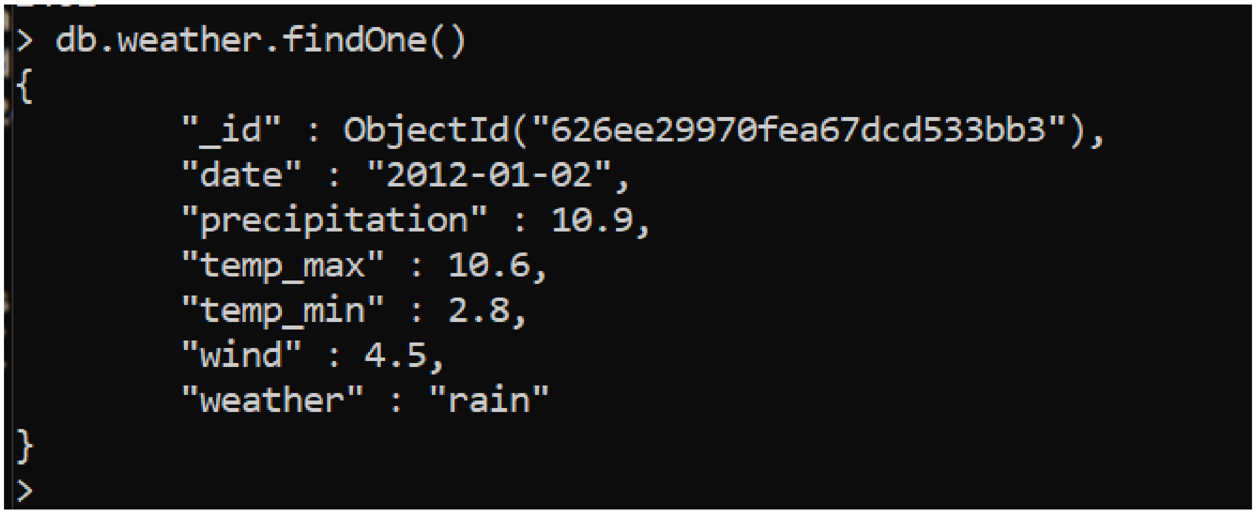


```
> db.weather.count()
1461
> 
```

Figure 39 Number of documents

By using the “count ()” method, total number of datasets can be counted where in the above collection there is 1461 document are available in this weather collection.

Display only one document details

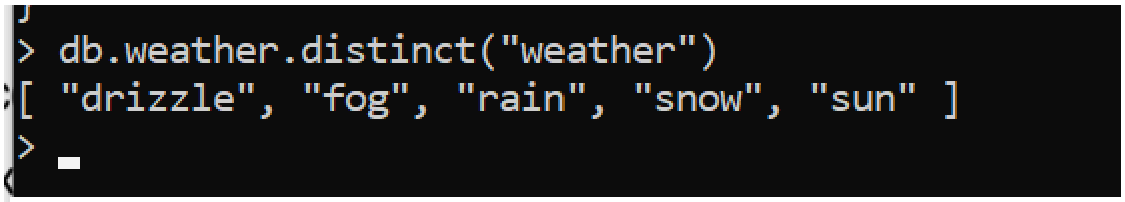


```
> db.weather.findOne()
{
        "_id" : ObjectId("626ee29970fea67dcd533bb3"),
        "date" : "2012-01-02",
        "precipitation" : 10.9,
        "temp_max" : 10.6,
        "temp_min" : 2.8,
        "wind" : 4.5,
        "weather" : "rain"
}
>
```

Figure 40 One Document Detail

To use “findOne()” method, only one document detail can be retrieve as shown in above figure.

Different class Data



```
> db.weather.distinct("weather")
[ "drizzle", "fog", "rain", "snow", "sun" ]
> 
```

Figure 41 Different cluster dataset

To display different type of class data “distinct ()” method is used where there are five type of cluster document are rendered.

Discussion of the findings

In the process of predicting weather, the data is required to make clean so that accuracy could increase. Visualization of the data is so important to understand the data like to find missing values, to find outliers to know how data is arranging. After cleaning data, there are various types of models/ algorithms to train data. For examples; Decision tree, linear regression, support vector machines etc. grid search CV play the important role to find best hyper parameter of the mode. It helps to find best hyper parameter, so that we can apply that model and hyper parameter to improve the accuracy. Elastic search analytical engine makes very easy to visualize the data. And the Apache spark helps to analyzed data quickly using queries.

Analysis of the findings

While visualizing the data, the library like matplotlib and seaborn was really helps to understand the data. Without cleaning data and then predicting weather makes low accuracy than after clean data. I have used 3 models i.e., logistic regression, random forest, and decision tree. Different models give different accuracy. Every model is best but the best accuracy gives according to the dataset. In the process of hyper parameter tuning, many hyper parameters can be added which can makes data output different accuracy. Hyperparameter tunning really help to find the best model for the prediction. Elastic search engine makes simple and easy to understand visualized data. And in Apache spark, using queries makes easy to analyses data.

Conclusion

In this report, there are a lot of technology is used to determined the prediction and forecasting the weather dataset such as AI based prediction system, Spark technology, Hadoop, MongoDB, and Elastic Search. Where by using the AI based model, it predicts the weather condition whether it rain, fuzzy, hot etc. And the other technology is use to forecasting the weather data and analyzed purpose. the weather prediction of the data using various models like, decision tree, logistic regression, and random forest regression. The weather predication's data contain 8 columns which "weather column" is target variable. This research has cover up about how big data is analyzed, how search engine (elastic search) is used to visualize, how Jupiter notebook and its libraries is used, and how Apache spark is used. It also talks about why do we clean data? Why do we visualize the data? Why do we use models? While predicting weather this research has used libraries NumPy and pandas to read and operate data, matplotlib, seaborn to visualize data, missing to find missing values, label to convert categorical to numerical values etc. the 3 model were used and hyper tuning is applied to find the best modes for the prediction. Big data is the data which contains the all type of data such as relational data, non-relational data and undefined data also that means unshaped data in any kind of formation which is quite difficult to measure these kinds of data and define the own mechanism to predict their future. Data mining is the option to process the weather data, calculate certain mechanism and predict the value forecasting using the machine learning.

References

A. K. Pandey, C. P. (2017). A hadoop based weather prediction model for classification of weather data. *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)* (pp. 1,2). Coimbatore, India: IEEE.

Chernyshov, K. (2005). Theses on entropy. *Proceedings. 2005 International Conference Physics and Control, 2005.* (p. 3). St. Petersburg, Russia: IEEE.

Comparative Analysis of Hadoop Tools and Spark Technology. (2018). *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (p. 2). Pune, India: IEEE.

Eugen, M. (2021). *Design of Short-Term Wind Production Forecasting Model using Machine Learning Algorithms.* Bucharest, Romania: IEEE.

habilelabs. (2022, 05 01). *How to Import CSV files in MongoDB Database - Habilelabs*. Retrieved from habilelabs: https://www.habilelabs.io/importing-csv-files-mongodb-database/

Ismail, K. A. (2016). *Big Data prediction framework for weather Temperature based on MapReduce algorithm.* Langkawi, Malaysia: IEEE.

John K. Williams* and D. A. Ahijevych, C. J. (2020). A MACHINE LEARNING APPROACH TO FINDING WEATHER REGIMES AND SKILLFUL PREDICTOR COMBINATIONS FOR SHORT-TERM STORM FORECASTING. *SKILLFUL PREDICTOR COMBINATIONS FOR SHORT-TERM STORM FORECASTING* (pp. 3,4). Boulder, Colorado: National Center for Atmospheric Research.

Kumar, R. (2022). Decision Tree for the Weather Forecasting. *International Journal of Computer Applications (0975 – 8887)*, 32-33.

Lu Han, L. Z. (2020). Design and Implementation of Elasticsearch for Media Data. *2020 International Conference on Computer Engineering and Application (ICCEA)*, (p. 1). Guangzhou, China.

Marius Eugen ȚIBOACĂ, S. C. (2021). Design of Short-Term Wind Production Forecasting Model using Machine Learning Algorithms. *2021 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE)* (pp. 2,3). Bucharest, Romania: IEEE.

Pwint Phyu Khine, W. Z. (2017). Big Data for Organizations: A Review. *Journal of Computer and Communications*, 1.

Sharma, A. (2019). Weather Forecasting Application using Linear and Logistic. *Proceedings of IOE Graduate Conference, 2019-Winter* (p. 252). Chyasal, Lalitpur, Nepal: Himalaya College of Engineering, Tribhuvan University, Institute of Engineering.

Shubham Madan, T. C. (2018). Analysis of Weather Prediction using Machine Learning & Big Data. *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)* (p. 1). Paris, France: IEEE.

Wofford, M. F. (2019). Jupyter Notebooks as Discovery Mechanisms for Open Science: Citation Practices in the Astronomy Community. *Computing in Science & Engineering (Volume: 22, Issue: 1, Jan.-Feb. 1 2020)* (p. 3). IEEE.

Zhi, T. (2018). A Gini Impurity-Based Interest Flooding Attack Defence Mechanism in NDN. *IEEE Communications Letters (Volume: 22, Issue: 3, March 2018)* (p. 3). IEEE.

Appendix

Work Distribution

Pramod Chaudhary (2050114)

1. Background of Weather Forecasting
2. Methodology
3. Result and Discussion
    a. Mongo DB
    a. Hadoop
    b. Discussion of findings
4. Conclusion

Yogesh Shrestha (2050214)

2. Abstracts
3. Related Work
4. Methodology (block diagram explanations)
5. Result and Discussion
    a. Using Jupyter notebook
    b. Using Elastic Search
    c. Using Spark
    d. Analysis of findings