INTRODUCTION

In the most recent century, weather forecasting has been one of the most challenging technological and experimental problems on the planet. Because of the abrupt changes that occur. It becomes challenging to anticipate the weather in the present with accuracy due to a number of restrictions in improving weather forecasting. In meteorology, weather forecasting plays a big part. Making an accurate prediction is one of the major challenges facing meteorologists anywhere in the world. Because they are used to protect lives and property, weather warnings are important. Utility companies use temperature projections to evaluate demand over the next few days. Estimates can be used to plan activities around these events, as well as to prepare for and survive them, because heavy rain, snow, and wind chill substantially reduce outdoor activities. Without accurate weather forecasts, people would find themselves in dangerous situations for which they were unprepared and suffer damage or worse. Among other things, understanding weather representation from a huge amount of meteorological datasets is one of the challenges of weather forecasting. In this analysis, the features and their relationships with one another will be visualised, and the results will be classified as either a forecast of the likelihood of rain the following day or the conditions that will bring it about. These data are not from Rio de Janeiro, but it is a place that experiences severe rains every year, where landslides have previously claimed lives and possessions, and where machine learning has assisted in predicting disasters around the world.

OBJECTIVES

The earth it’s a very complex place suffering of the climate change, It’s significant to predict an accurate weather without any error, to make sure of the security and mobility, as well a safe daily operation. Weather forecasting it’s build by collecting huge amount of data, that’s make machine learning an essential tool, by using some back testing method and some algorithms to make an accurate prediction of weather. This project is going to evaluate the methods with a set of experiments that highlight the performance and value of the methods. With ML add in weather forecasting bring an enormous advantage for the prediction make it more accurate. Weather forecasting it’s a very powerful application of science for the benefits of the society it’s can be used for Aerospatiale; agriculture; sport; musical event; marine; renewable energy; aviation and forestry, it’s very important to know what’s the weather going to be. Finding out the maximum and minimum temperature, it’s one of the method of linear regression that’s reach a precise result. In this project I will demonstrate how to build a model predicting weather analysis in Python.

LITERATURE REVIEW

Machine Learning is concerned with the design and development of algorithms that allow computers to evolve behaviours based on empirical data, such as from sensor data or databases. Predictability of weather with a numerical solution of statistics that control movement and climate change. Many climate forecasting techniques in [1] the author have used Decision trees to measure the Information gain from various predictors and the split was decided based on nodes having the highest information gain. Over-fitting of the data is prevented by pruning of the tree. By getting rid of nodes that do not contribute much to the information gain, pruning mechanism maximizes information gain. As a result the most effective predictors in a given data set are left behind. Support vector machine (SVM), it has been observed that generation of clusters impacts positively on the prediction of the in rainfall. Results have been presented through statistical and graphical analyses. Behaviour of systems with many interdependent components that lead to organized as well as irregular features is referred to as complexity. In such systems the knowledge of the parts does not necessarily lead to the predictable behaviour of the entire system. Complexities associated with meteorological and geophysics processes have been reviewed in Sharma et al (2012). Many researchers recommended ensemble forecasts to improve forecast accuracy of the models [5]. Sandeep et all [6] recommended that there is a need for emerging techniques such as machine learning [7], deep learning and IoT based weather monitoring system [8]-[9].

SYSTEM SPECIFICATON

System configuration mainly refers to the specification of a given computer system, from its hardware components to the software and various processes that are run within that system. It refers to what types and models of devices are installed and what specific software is being used to run the various parts of the computer system.

- Windows: 10 or newer
- MAC: OS X v10.7 or higher

SOFTWARE SPECIFICATION

perating system	'indows 10
oding language	ython language
oftware tool	oogle Colab, MS office

HARDWARE SPECIFICATION

We strongly recommend a computer fewer than 5 years old.

- Processor: Minimum 1 GHz; Recommended 2GHz or more
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
- Hard Drive: Minimum 32 GB; Recommended 64 GB or more
- Memory (RAM): Minimum 1 GB; Recommended 4 GB or above
- Sound card w/speakers
- Some classes require a camera and microphone

Processor	Intel(R) Celeron(R) CPU J3060 @ 1.60GHz 1.60GHz
Hard disk	1TB
RAM	4GB

DATASET

- **Temperature -** The measure of warmth or coldness.
- **Humidity -** The amount of moisture in the atmosphere.
- **Rainfall -** The amount of moisture (usually rain or snow) which falls on the ground.
- **Wind Speed -** The speed at which air flows through the environment.
- **Wind Direction** – The direction in which the wind is moving.
- **Pressure -** The force the atmosphere applies on the environment.
- **Sunshine-** The amount of sunlight is dependent on the extent of the daytime cloud cover.

METHODOLOGY

This figure shows the working methodology of Weather analysis and predictions using machine learning.

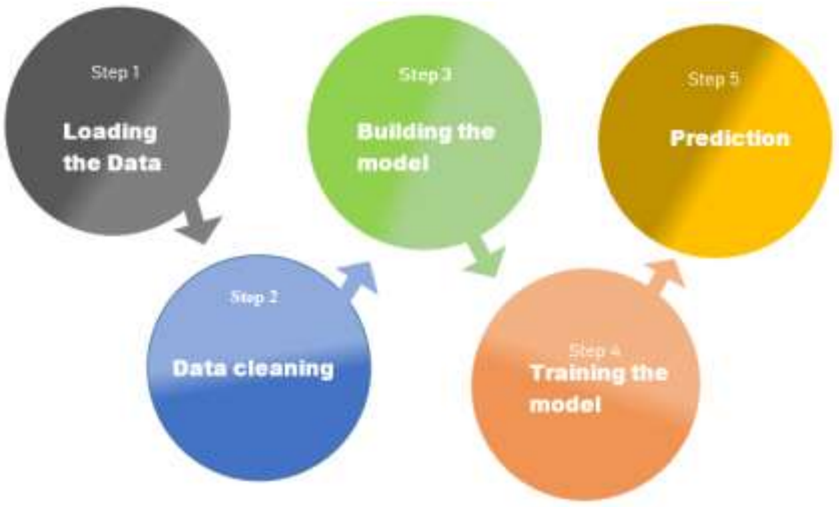


Figure 1: Working mechanism

Data Pre-Processing:

Pre-processing involves a number of procedures that make the collected data more logical. Pre-processing involves removing extraneous and missing values from the data.

Filling Missing Values

There are several ways to treat missing data. Each with its own strengths and weaknesses. A common strategy is to fill the data with the averages but it is usually not as close to the real value as if I used KNN, it is a good solution for small datasets.

KNN imputer

The KNN Imputer is used to fill in missing values in a dataset using the k-Nearest Neighbours method. K-Nearest Neighbours algorithm is used for classification and prediction problems. The KNN Imputer predicts the value of a missing value by observing trends in related columns.

Imputation for completing missing values using k-Nearest Neighbours. Each sample's missing values are imputed using the mean value from n_neighbours nearest neighbours found in the training set. Two samples are close if the features that neither is missing are close.

Training the Model

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model. This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model. Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved.

Scikit-Learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including Classification, Regression, Clustering and dimensionality reduction via a consistence interface in Python.

Machine Learning Algorithm’s

Decision Tree Classifier

Decision tree classifiers are supervised machine learning models. This means that they use prelabelled data in order to train an algorithm that can be used to make a prediction. Decision trees can also be used for regression problems. Much of the information that youll learn in this tutorial can also be applied to regression problems.Decision tree classifiers work like flowcharts. Each node of a decision tree represents a decision point that splits into two leaf nodes. Each of these nodes represents the outcome of the decision and each of the decisions can also turn into decision nodes. Eventually, the different decisions will lead to a final classification.

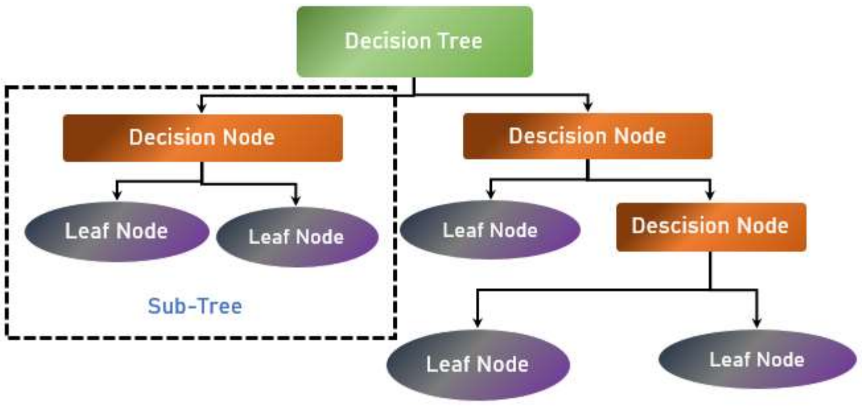


Figure2: Decision tree clasifier

Random Forests

The random forests algorithm is an ensemble method which constructs decision trees based on a bootstrapped sample from the dataset. The features for each split point in the decision tree are generated using a bootstrapped set of the features of the overall dataset, and the bootstrapped set is then used to determine the optimal feature to be used for the decision rule.

This process generates an individual decision tree classifier, which, due to the use of bootstrapping, is a model that has low bias, but high variance. Thus, by aggregating predictions across all trees, the random forests algorithm can reduce the overall variance of the model, while still keeping bias relatively low.

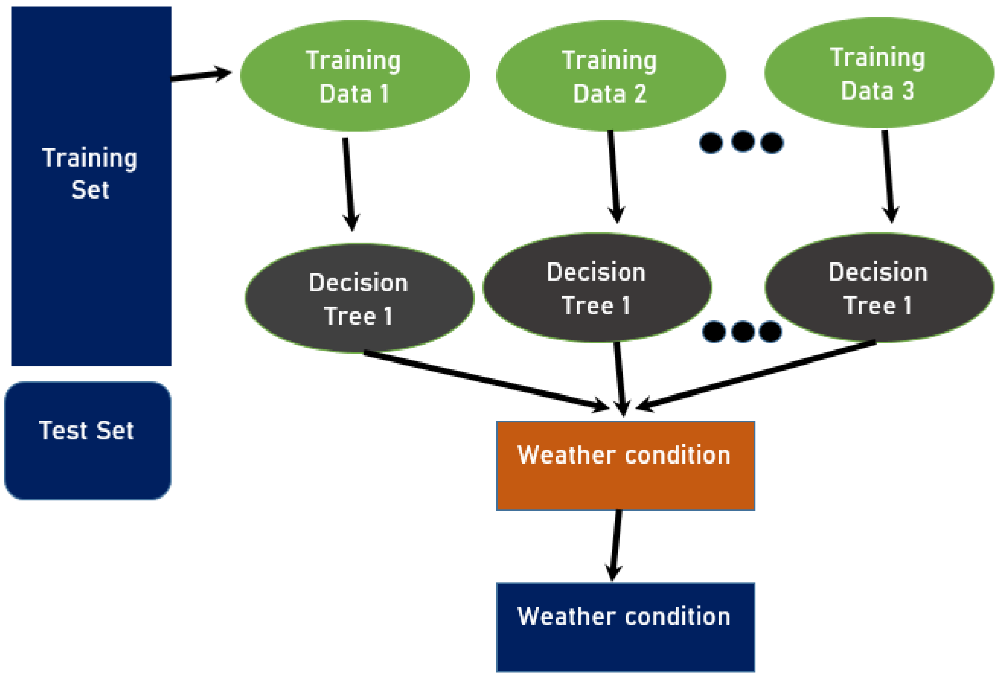


Figure3: Random forest algorithm

Support Vector Classifier

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.

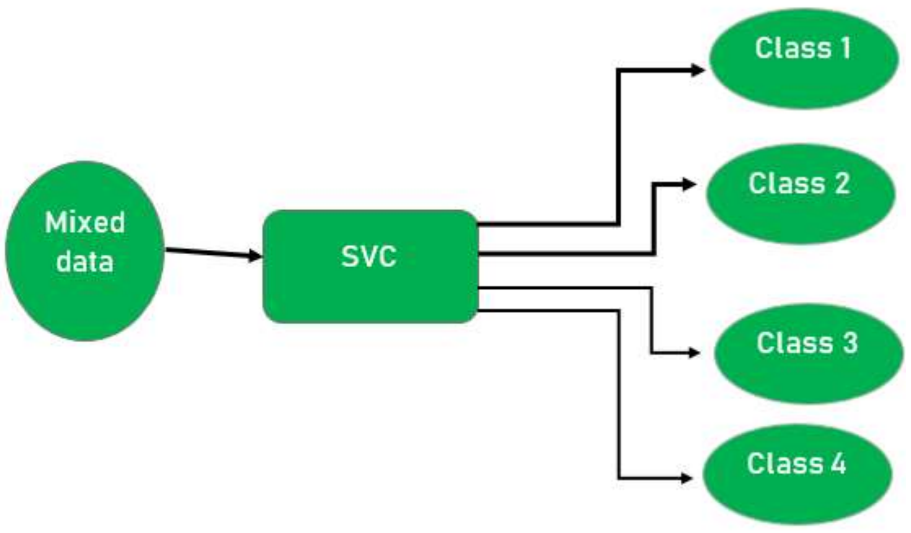


Figure4: Support vector classifier

Logistic Regression

The binomial logistic regression algorithm works by finding the theta value that maximizes the following log likelihood function: where x represents the features matrix, y represents the labels vector, and h(x) is 1/(1 + exp(− θ x)

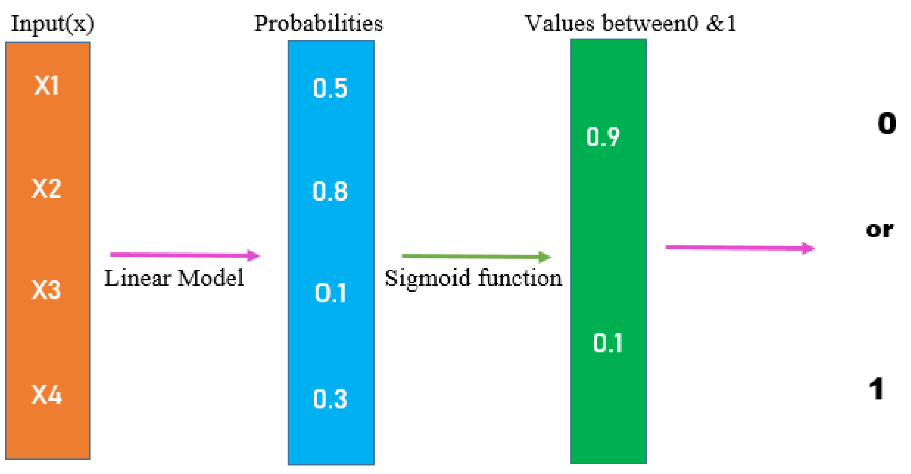


Figure5: Logistic regression

Prediction

A prediction is a guess about what might happen in the future, based on observations that make. Predicting is closely related to other process skills such as observing, inferring, and classifying. Many process skills are in fact dependent on other process skills.

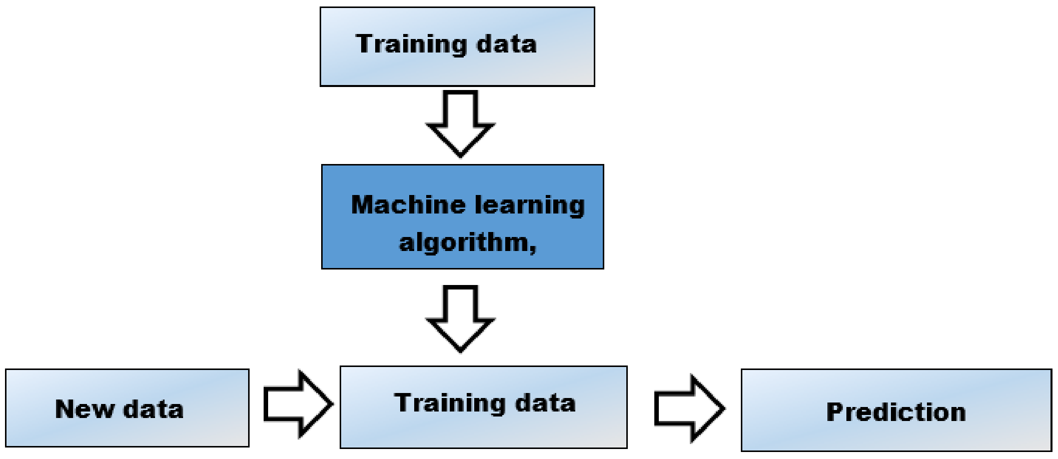


Figure6: Prediction process

Confusion Matrix

A model's performance on a particular dataset is described using a confusion matrix, which is a summary of predictions in the form of a table that provides answers to classification problems. With count values, the accurate and incorrect predictions are tallied up and broken out by each class. This is the confusion matrix's secret. The confusion matrix demonstrates the manner in which your classification model makes predictions while being confused. The Confusion matrix provides further information about the types of errors that are being produced in addition to providing insight into the mistakes that a classifier is making.

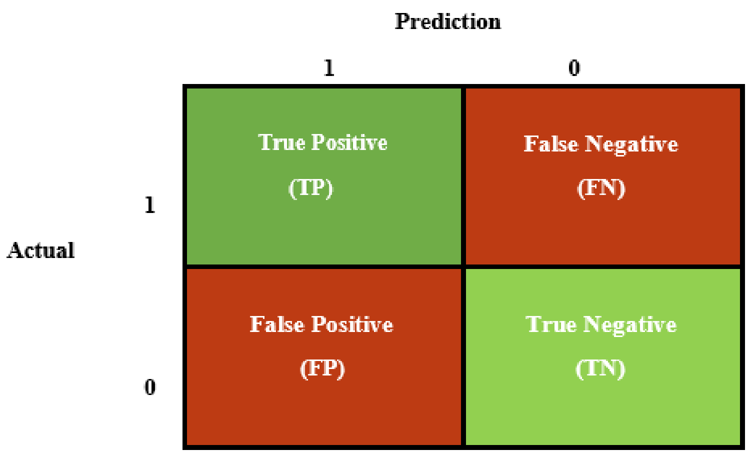
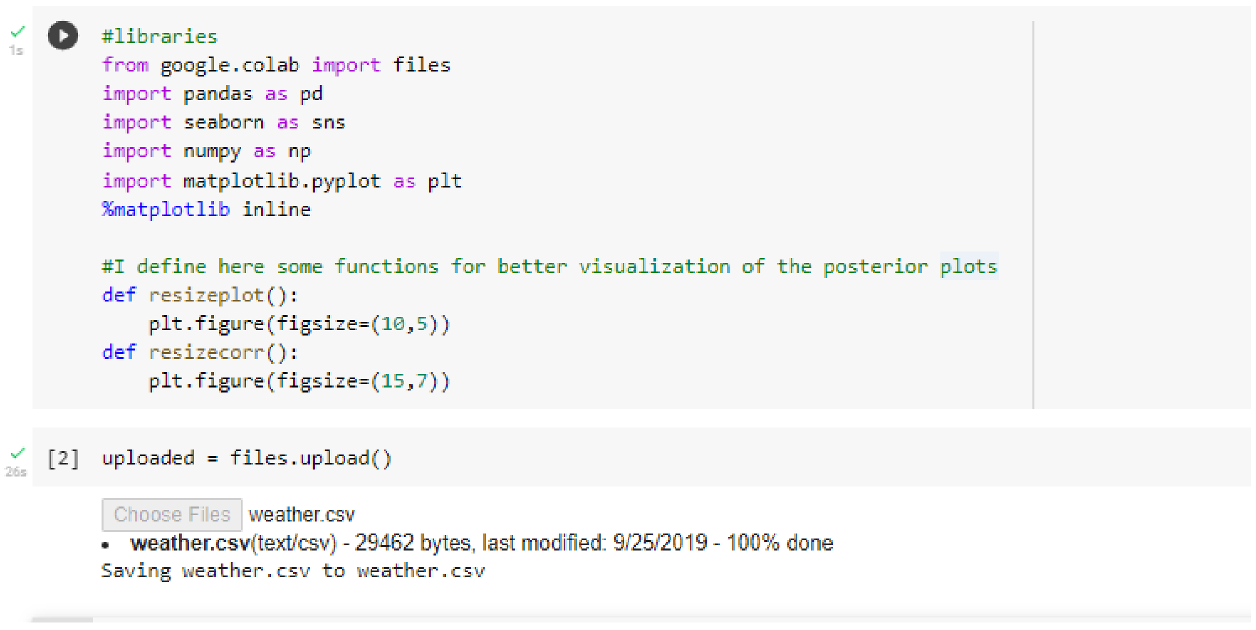


Figure 7: Confusion Matrix

RESULTS AND DISCUSSION

Load the Dataset

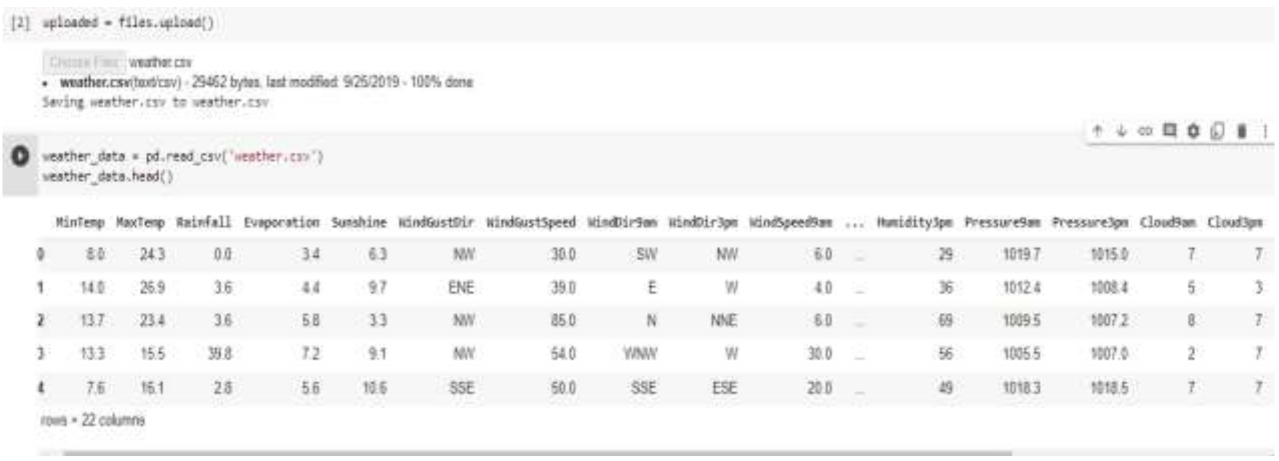
- First import the libraries.
- I define here some functions for better visualization of the posterior plots.
- Load the data:



```
#libraries
from google.colab import files
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

#I define here some functions for better visualization of the posterior plots
def resizeplot():
    plt.figure(figsize=(10,5))
def resizecorr():
    plt.figure(figsize=(15,7))
```

```
[2] uploaded = files.upload()
```

```
Choose Files weather.csv
• weather.csv(text/csv) - 29462 bytes, last modified: 9/25/2019 - 100% done
Saving weather.csv to weather.csv
```



```
[2] uploaded = files.upload()
```

```
Choose Files weather.csv
• weather.csv(text/csv) - 29462 bytes, last modified: 9/25/2019 - 100% done
Saving weather.csv to weather.csv
```

```
weather_data = pd.read_csv('weather.csv')
weather_data.head()
```

```
   MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustDir  WindGustSpeed  WindDir9am  WindDir3pm  WindSpeed9am  ...  Humidity3pm  Pressure9am  Pressure3pm  Cloud9am  Cloud3pm
0      8.0     24.3       0.0          3.4       6.3           NW           30.0          SW          NW           6.0  ...           29       1019.7       1015.0         7         7
1     14.0     26.9       3.6          4.4       9.7          ENE           39.0           E           W           4.0  ...           36       1012.4       1008.4         5         3
2     13.7     23.4       3.6          5.8       3.3           NW           85.0           N         NNE           6.0  ...           69       1009.5       1007.2         8         7
3     13.3     15.5      39.8          7.2       9.1           NW           54.0         WNW           W          30.0  ...           56       1005.5       1007.0         2         7
4      7.6     16.1       2.8          5.6      10.6          SSE           50.0         SSE         ESE          20.0  ...           49       1018.3       1018.5         7         7
rows × 22 columns
```

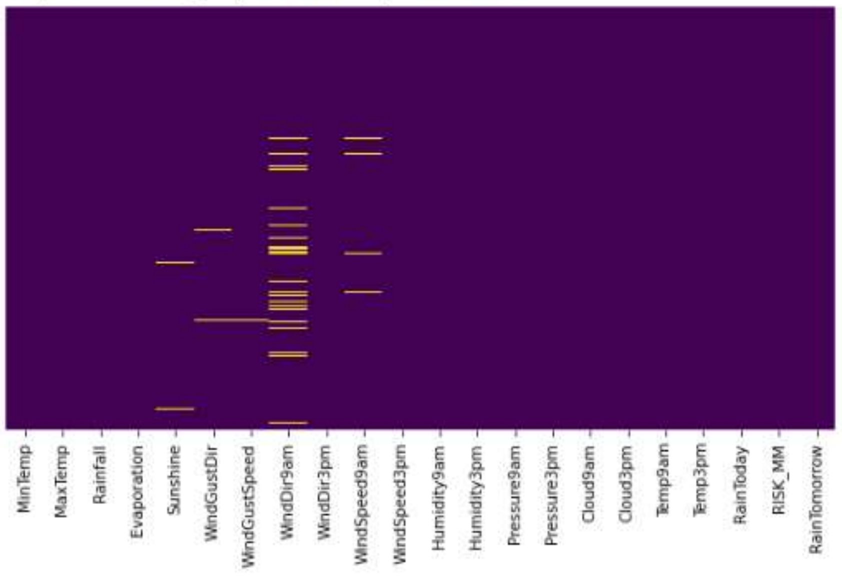
Data Pre-Processing:

```
[ ]  weather_data.isnull().sum()
```

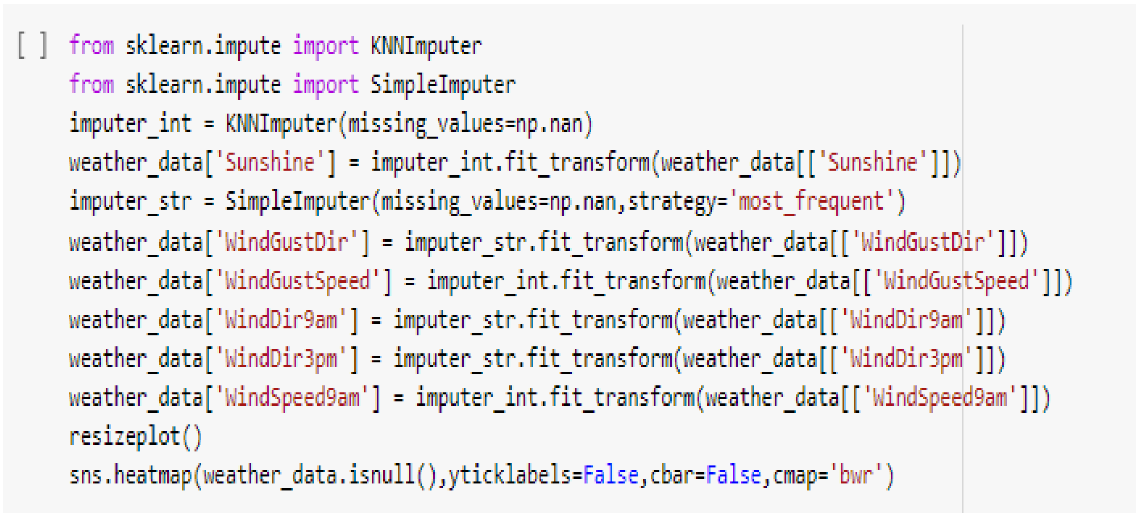
```
MinTemp           0
MaxTemp           0
Rainfall          0
Evaporation       0
Sunshine          3
WindGustDir       3
WindGustSpeed     2
WindDir9am       31
WindDir3pm        1
WindSpeed9am      7
WindSpeed3pm      0
Humidity9am       0
Humidity3pm       0
Pressure9am       0
Pressure3pm       0
Cloud9am          0
Cloud3pm          0
Temp9am           0
Temp3pm           0
RainToday         0
RISK_MM           0
RainTomorrow      0
dtype: int64
```

```
[ ]  resizeplot()
     sns.heatmap(weather_data.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f053ce23290>
```

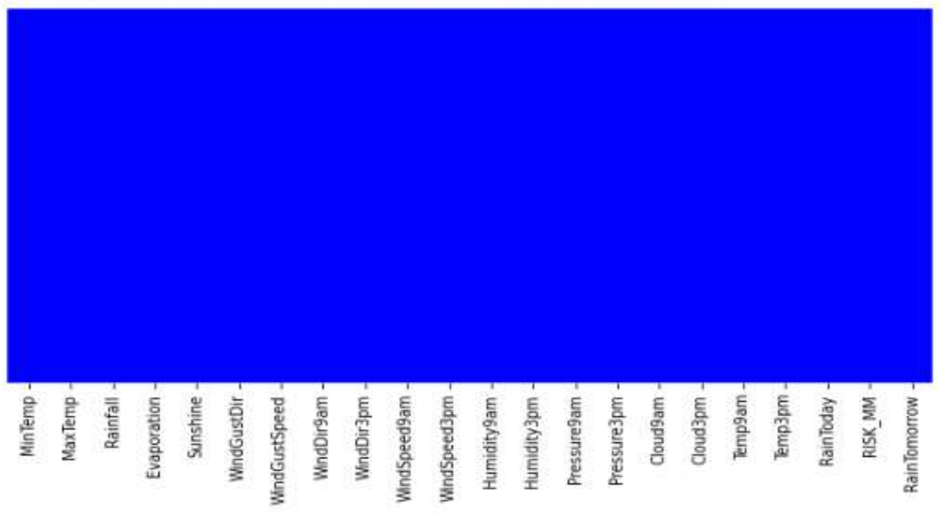


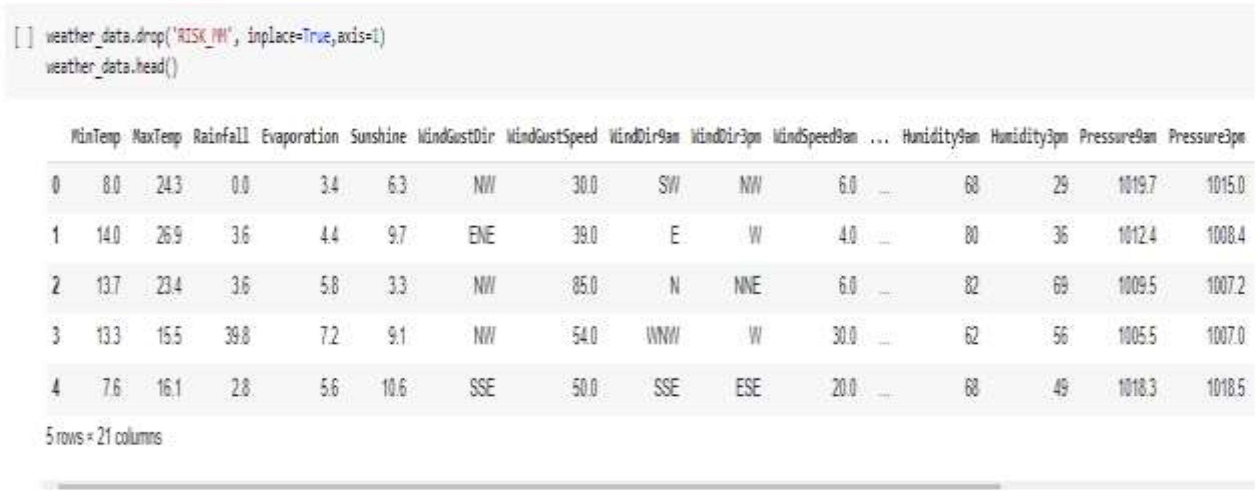
Filling Missing Values



```
[ ] from sklearn.impute import KNNImputer
    from sklearn.impute import SimpleImputer
    imputer_int = KNNImputer(missing_values=np.nan)
    weather_data['Sunshine'] = imputer_int.fit_transform(weather_data[['Sunshine']])
    imputer_str = SimpleImputer(missing_values=np.nan,strategy='most_frequent')
    weather_data['WindGustDir'] = imputer_str.fit_transform(weather_data[['WindGustDir']])
    weather_data['WindGustSpeed'] = imputer_int.fit_transform(weather_data[['WindGustSpeed']])
    weather_data['WindDir9am'] = imputer_str.fit_transform(weather_data[['WindDir9am']])
    weather_data['WindDir3pm'] = imputer_str.fit_transform(weather_data[['WindDir3pm']])
    weather_data['WindSpeed9am'] = imputer_int.fit_transform(weather_data[['WindSpeed9am']])
    resizeplot()
    sns.heatmap(weather_data.isnull(),yticklabels=False,cbar=False,cmap='bwr')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f053a514dd0>
```



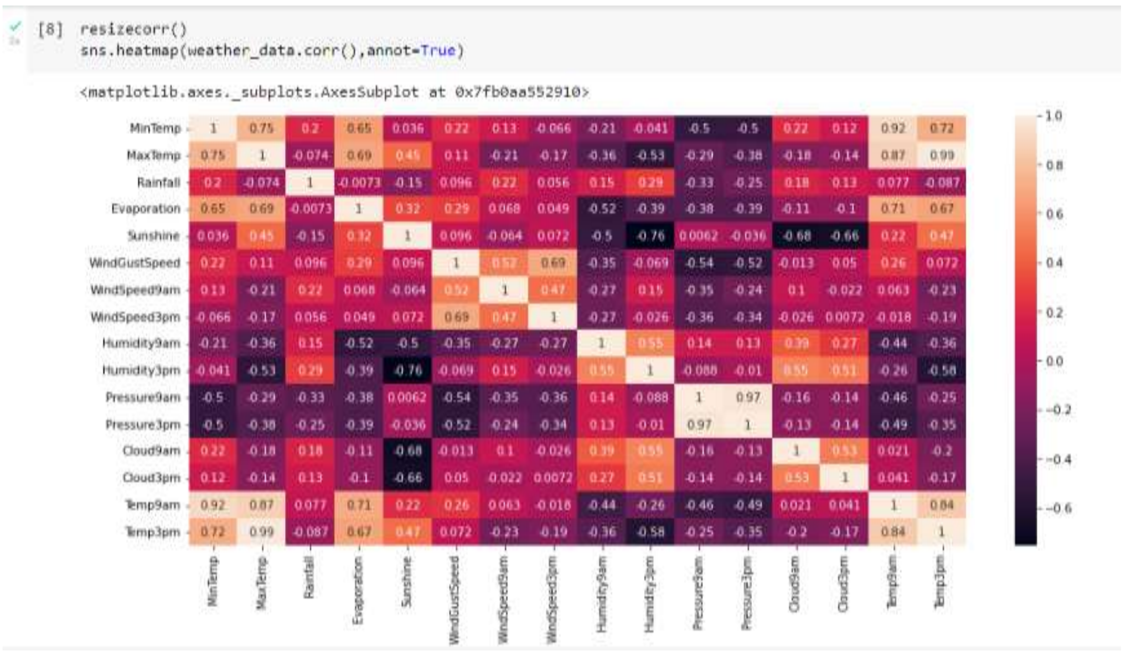


```
[ ] weather_data.drop('RISK_MM', inplace=True,axis=1)
    weather_data.head()
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm
0	8.0	24.3	0.0	3.4	6.3	NW	30.0	SW	NW	6.0	...	68	29	1019.7	1015.0
1	14.0	26.9	3.6	4.4	9.7	ENE	39.0	E	W	4.0	...	80	36	1012.4	1008.4
2	13.7	23.4	3.6	5.8	3.3	NW	85.0	N	NNE	6.0	...	82	69	1009.5	1007.2
3	13.3	15.5	39.8	7.2	9.1	NW	54.0	WNW	W	30.0	...	62	56	1005.5	1007.0
4	7.6	16.1	2.8	5.6	10.6	SSE	50.0	SSE	ESE	20.0	...	68	49	1018.3	1018.5

5 rows × 21 columns

This is a section that reveals a lot about the dataset. The correlation graph demonstrates the connection between the characteristics and how they affect others

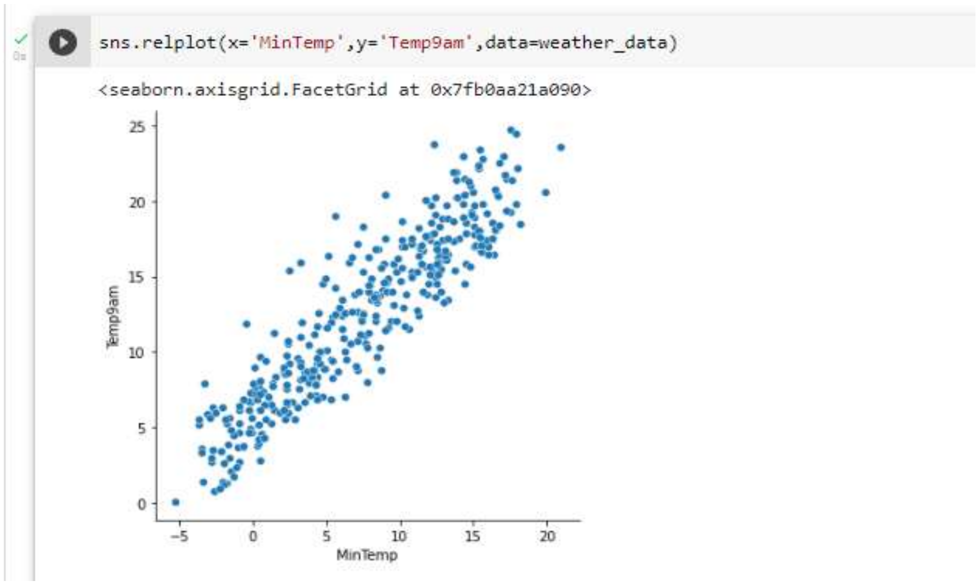


```
[8] resizecorr()
    sns.heatmap(weather_data.corr(),annot=True)

<matplotlib.axes._subplots.AxesSubplot at 0x7fb0aa552910>
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm
MinTemp	1	0.75	0.2	0.65	0.036	0.22	0.13	-0.066	-0.21	-0.041	-0.5	-0.5	0.22	0.12	0.92	0.72
MaxTemp	0.75	1	-0.074	0.69	0.45	0.11	-0.21	-0.17	-0.36	-0.53	-0.29	-0.38	-0.18	-0.14	0.87	0.99
Rainfall	0.2	-0.074	1	-0.0073	-0.15	0.096	0.22	0.056	0.15	0.29	-0.33	-0.25	0.18	0.13	0.077	-0.087
Evaporation	0.65	0.69	-0.0073	1	0.32	0.29	0.068	0.049	-0.52	-0.39	-0.38	-0.39	-0.11	-0.1	0.71	0.67
Sunshine	0.036	0.45	-0.15	0.32	1	0.096	-0.064	0.072	-0.5	-0.76	0.0062	-0.036	-0.68	-0.66	0.22	0.47
WindGustSpeed	0.22	0.11	0.096	0.29	0.096	1	0.52	0.69	-0.35	-0.069	-0.54	-0.52	-0.013	0.05	0.26	0.072
WindSpeed9am	0.13	-0.21	0.22	0.068	-0.064	0.52	1	0.47	-0.27	0.15	-0.35	-0.24	0.1	-0.022	0.063	-0.23
WindSpeed3pm	-0.066	-0.17	0.056	0.049	0.072	0.69	0.47	1	-0.27	-0.026	-0.36	-0.34	-0.026	0.0072	-0.018	-0.19
Humidity9am	-0.21	-0.36	0.15	-0.52	-0.5	-0.35	-0.27	-0.27	1	0.55	0.14	0.13	0.39	0.27	-0.44	-0.36
Humidity3pm	-0.041	-0.53	0.29	-0.39	-0.76	-0.069	0.15	-0.026	0.55	1	-0.088	-0.01	0.55	0.51	-0.26	-0.58
Pressure9am	-0.5	-0.29	-0.33	-0.38	0.0062	-0.54	-0.35	-0.36	0.14	-0.088	1	0.97	-0.16	-0.14	-0.46	-0.25
Pressure3pm	-0.5	-0.38	-0.25	-0.39	-0.036	-0.52	-0.24	-0.34	0.13	-0.01	0.97	1	-0.13	-0.14	-0.49	-0.35
Cloud9am	0.22	-0.18	0.18	-0.11	-0.68	-0.013	0.1	-0.026	0.39	0.55	-0.16	-0.13	1	0.53	0.021	-0.2
Cloud3pm	0.12	-0.14	0.13	-0.1	-0.66	0.05	-0.022	0.0072	0.27	0.51	-0.14	-0.14	0.53	1	0.041	-0.17
Temp9am	0.92	0.87	0.077	0.71	0.22	0.26	0.063	-0.018	-0.44	-0.26	-0.46	-0.49	0.021	0.041	1	0.84
Temp3pm	0.72	0.99	-0.087	0.67	0.47	0.072	-0.23	-0.19	-0.36	-0.58	-0.25	-0.35	-0.2	-0.17	0.84	1

Build The Model

The graph above demonstrates a strong link between "MinTemp" and "Temp9am," which is consistent with the fact that mornings are when temperatures are at their lowest.

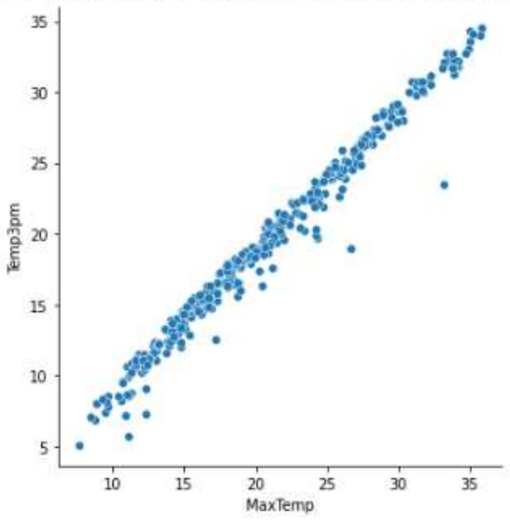


```
sns.relplot(x='MinTemp',y='Temp9am',data=weather_data)

<seaborn.axisgrid.FacetGrid at 0x7fb0aa21a090>
```

I must admit that I think a graph showing the link between two qualities that almost perfectly relate is beautiful. The scenario is the same as in the previous paragraph, but with the features "MaxTemp" and "Temp3pm," which also make sense because higher temperatures occur after 12:00h.

```
[ ]  sns.relplot(x='MaxTemp',y='Temp3pm',data=weather_data)
```

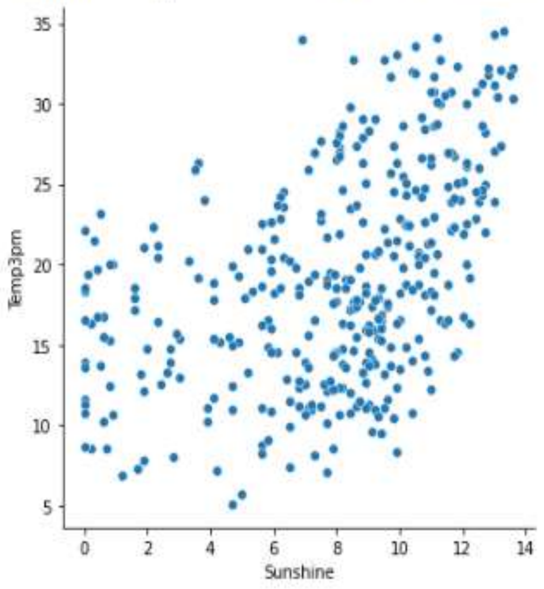
```
<seaborn.axisgrid.FacetGrid at 0x7f053749de10>
```



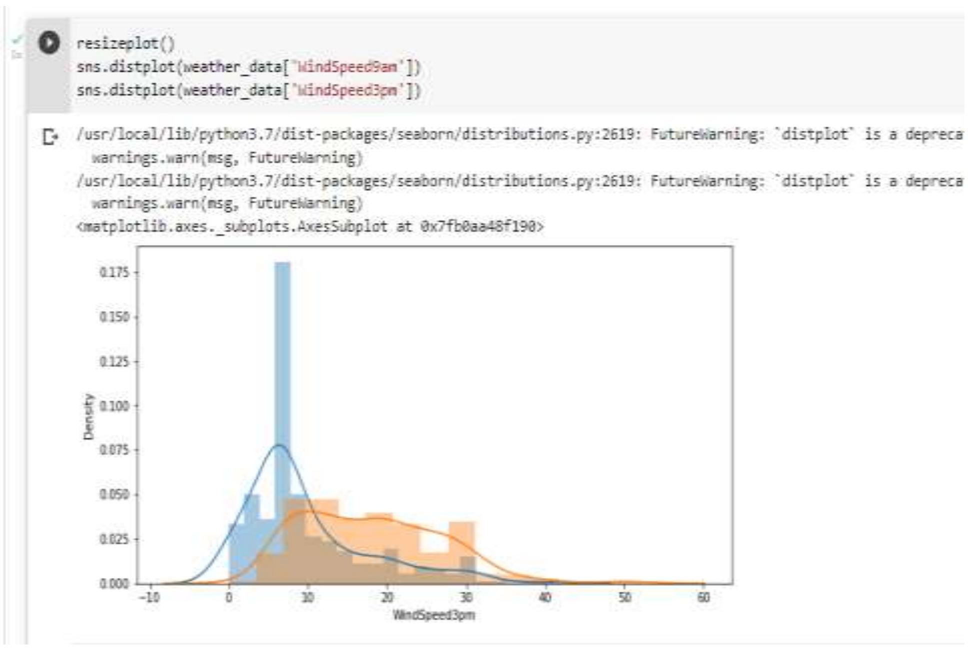
The relationship between Sunshine and Temp3pm is depicted in the graph below, which would seem to indicate that there would be a very strong correlation, but as we can see, there isn't one. This can be caused by a number of variables, such as the amount of clouds in the afternoon when solar illumination is low.

```
[ ]  sns.relplot(x='Sunshine',y='Temp3pm',data=weather_data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f053745e910>
```



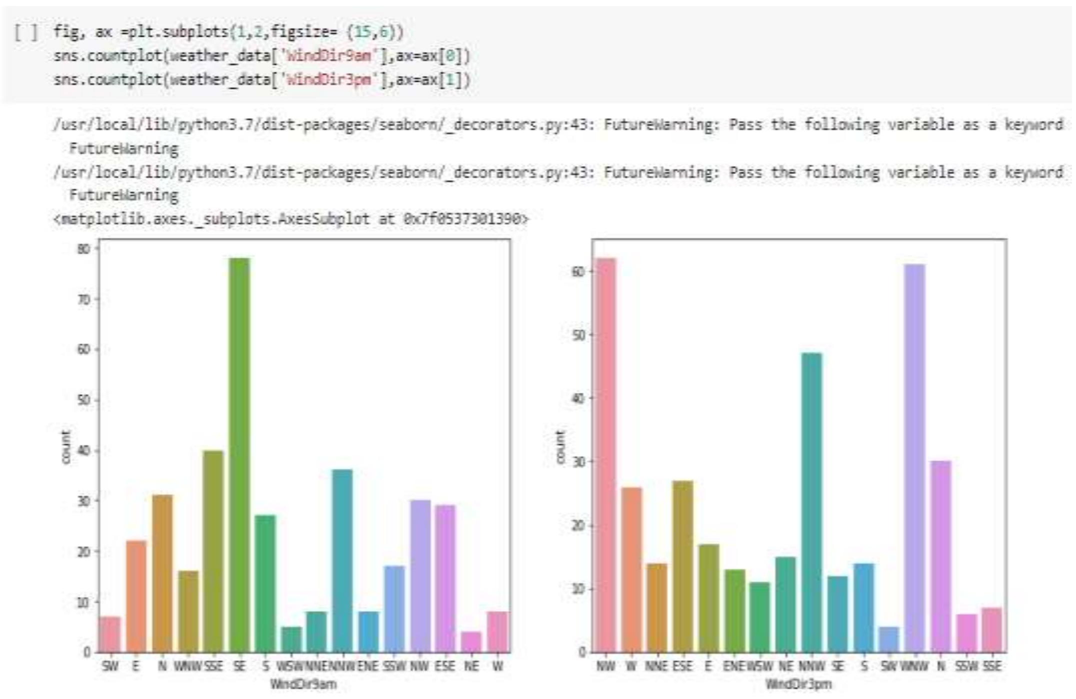
Below is a comparison of two features, the wind speed at two times, 9:00 am and 3:00 pm



```
resizeplot()
sns.distplot(weather_data['WindSpeed9am'])
sns.distplot(weather_data['WindSpeed3pm'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a depreca
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a depreca
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fb0aa48f190>
```

Two features that are particularly difficult to predict or mimic are shown below. There are a variety of differences and amplitudes in the wind directions in both plots, which also supports the notion that these two variables are influenced by elements such as speed variation with height, the presence of obstructions in the immediate vicinity, relief that can have an impact on air flow acceleration or deceleration, etc.

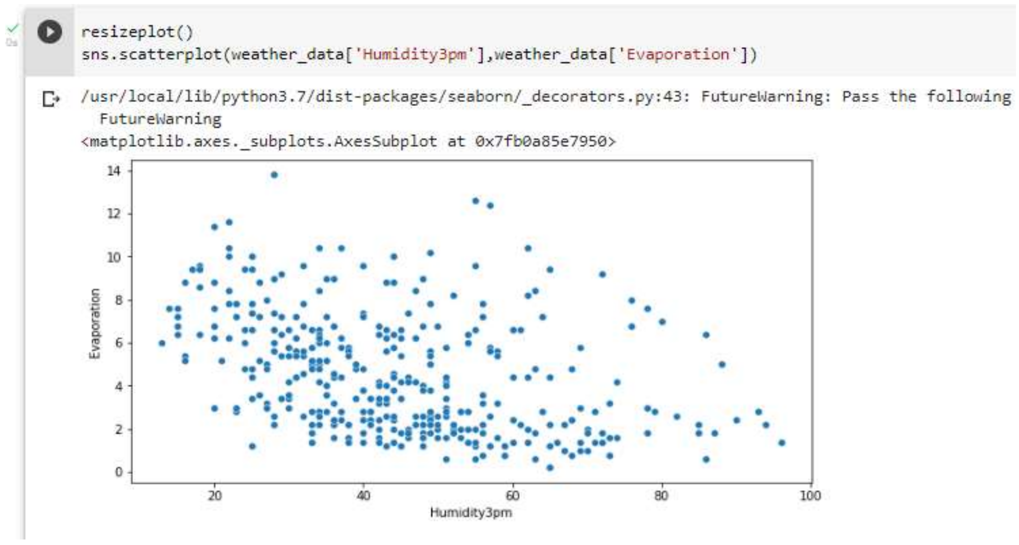


```
[ ] fig, ax =plt.subplots(1,2,figsize= (15,6))
    sns.countplot(weather_data['WindDir9am'],ax=ax[0])
    sns.countplot(weather_data['WindDir3pm'],ax=ax[1])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f0537301390>
```

count
SW E N WNW SSE SE S WSW NNE NNW ENE SSW NW ESE NE W
WindDir9am
count
NW W NNE ESE E ENE WSW NE NNW SE S SW WNW N SSW SSE
WindDir3pm

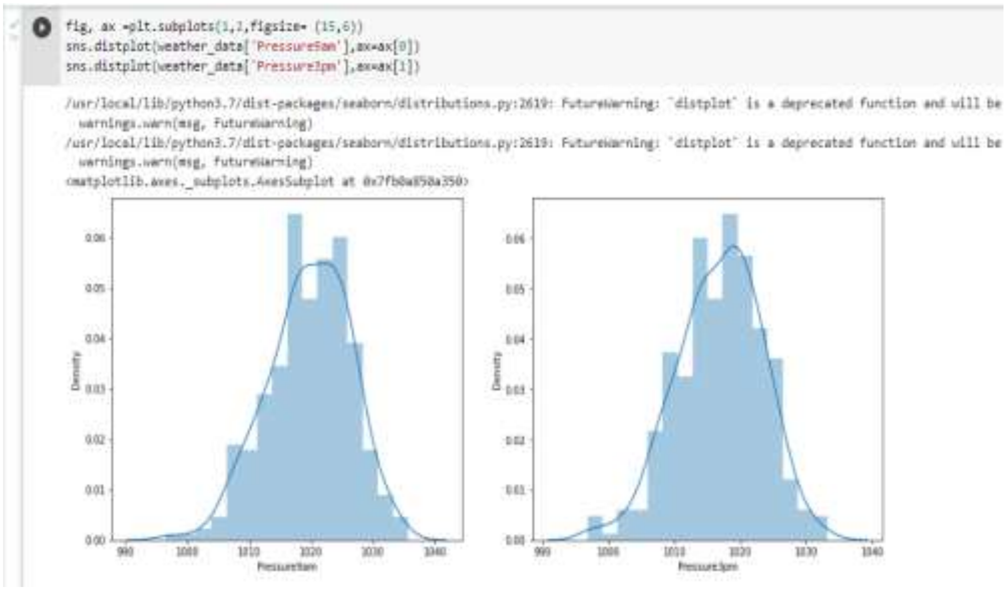
Technically speaking, the less water vapour there is in the air, the easier it will be for the water to evaporate. In the most extreme circumstance, where the relative humidity is 100%, the same amount of water that evaporates from the water condenses in it, leading to a net zero evaporation rate.



```
resizeplot()
sns.scatterplot(weather_data['Humidity3pm'],weather_data['Evaporation'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fb0a85e7950>
```

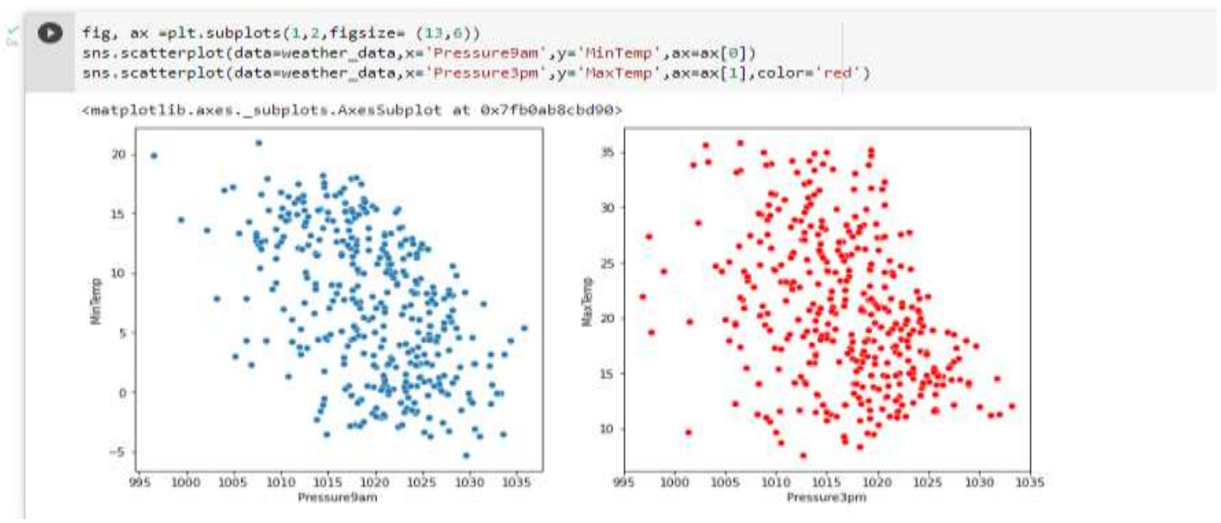
I noticed a lot of identical data in this distribution graph, which suggests that the temperatures influencing the pressure did not change significantly during the day.



```
fig, ax =plt.subplots(1,2,figsize= (15,6))
sns.distplot(weather_data['Pressure9am'],ax=ax[0])
sns.distplot(weather_data['Pressure3pm'],ax=ax[1])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fb0a858a350>
```

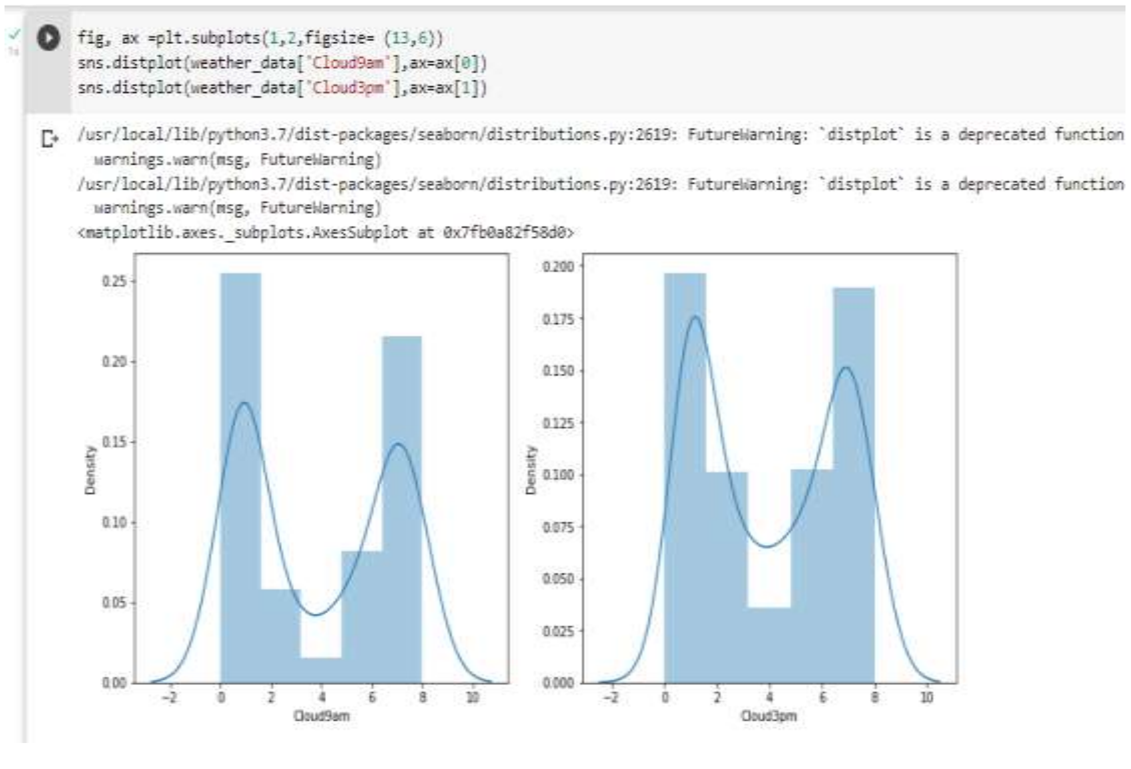
This scatter plot demonstrates that the relationship between pressure in the morning and afternoon is relatively similar, but it is also clear that the dispersion is denser in some variations in the afternoon (3pm). That clarifies the distplot seen above.



```
fig, ax =plt.subplots(1,2,figsize= (13,6))
sns.scatterplot(data=weather_data,x='Pressure9am',y='MinTemp',ax=ax[0])
sns.scatterplot(data=weather_data,x='Pressure3pm',y='MaxTemp',ax=ax[1],color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb0ab8cbd90>
```

The very similarity of the figures below suggests that at specific times, humidity and evaporation are comparable, which results in comparable cloud forms.

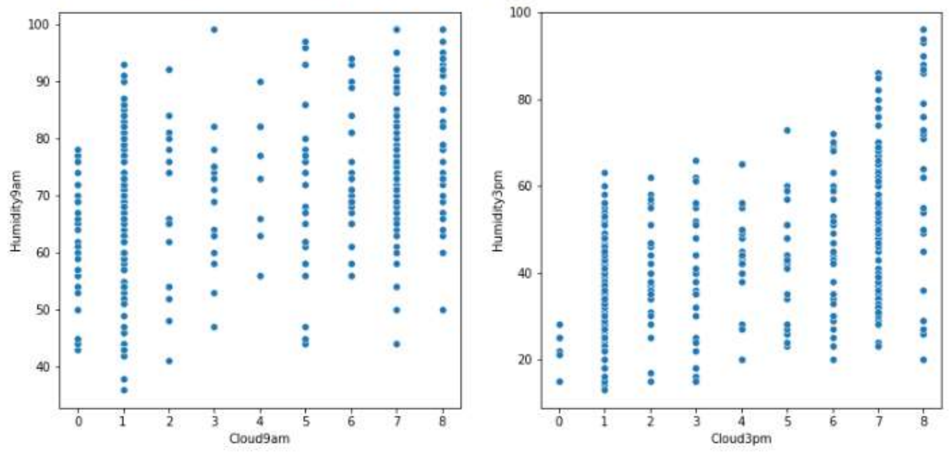


```
fig, ax =plt.subplots(1,2,figsize= (13,6))
sns.distplot(weather_data['Cloud9am'],ax=ax[0])
sns.distplot(weather_data['Cloud3pm'],ax=ax[1])
```

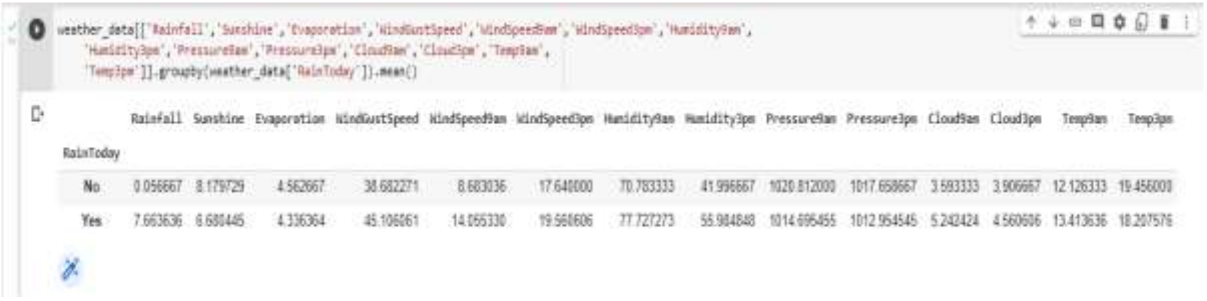
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fb0a82f58d0>
```

Here, I examine the connection between evaporation and cloud formation in more detail. Naturally, I have a greater rate in the morning because of the relative humidity, but in the afternoon, my rates are more consistently based on the time.

```
[ ] fig, ax =plt.subplots(1,2,figsize= (13,6))
    sns.scatterplot(data=weather_data, x='Cloud9am',y='Humidity9am',ax=ax[0])
    sns.scatterplot(data=weather_data,x='Cloud3pm',y='Humidity3pm',ax=ax[1])

<matplotlib.axes._subplots.AxesSubplot at 0x7f0536d9f990>
```

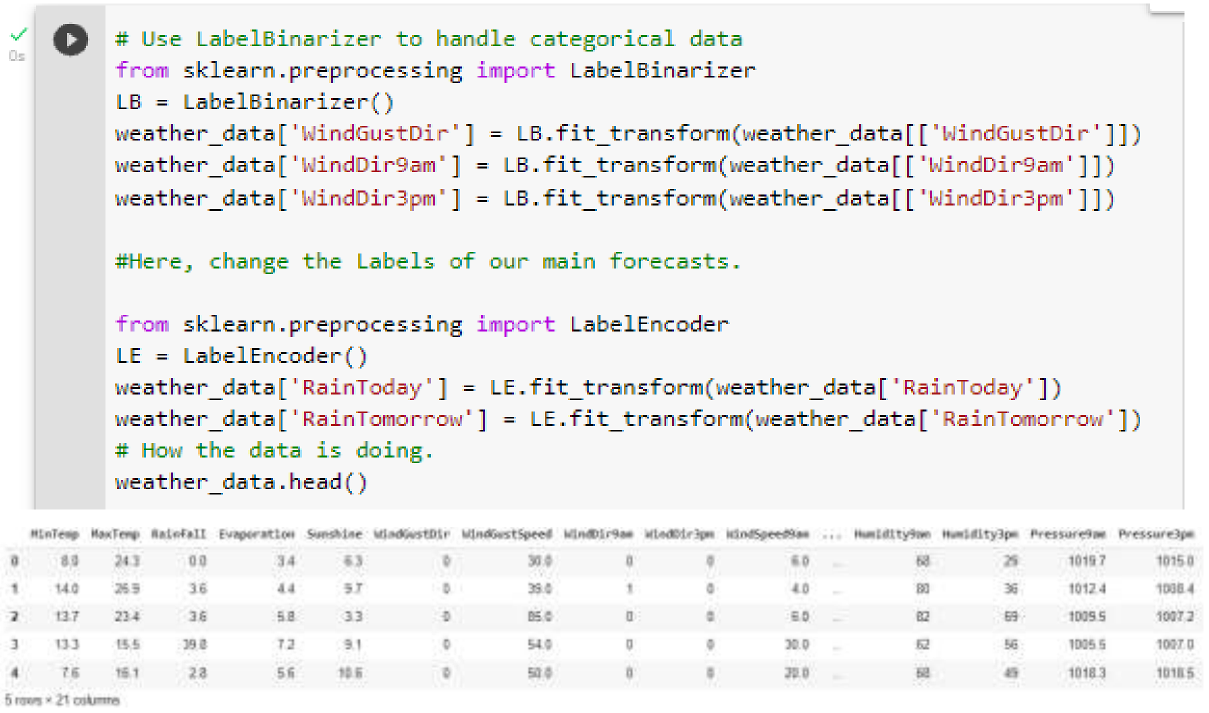


In this "groupby" we have important information about the possible conditions that may or may not cause rain conditions.



```
weather_data[['Rainfall','Sunshine','Evaporation','WindGustSpeed','WindSpeed9am','WindSpeed3pm','Humidity9am',
    'Humidity3pm','Pressure9am','Pressure3pm','Cloud9am','Cloud3pm','Temp9am',
    'Temp3pm']].groupby(weather_data['RainToday']).mean()

          Rainfall  Sunshine  Evaporation  WindGustSpeed  WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  Pressure3pm  Cloud9am  Cloud3pm    Temp9am    Temp3pm
RainToday
No        0.056667  8.179729     4.562667      38.682271      8.683036     17.640000    70.783333    41.996667  1020.812000  1017.658667  3.593333  3.906667  12.126333  19.456000
Yes       7.663636  6.680445     4.336364      45.106061     14.055330     19.560606    77.727273    55.984848  1014.695455  1012.954545  5.242424  4.560606  13.413636  18.207576
```

In this stage, categorical data will be handled using LabelBinarizer.

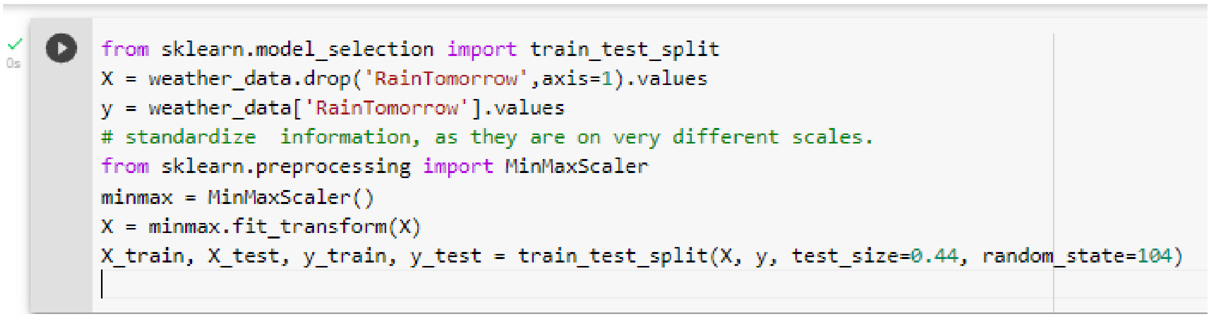


```
# Use LabelBinarizer to handle categorical data
from sklearn.preprocessing import LabelBinarizer
LB = LabelBinarizer()
weather_data['WindGustDir'] = LB.fit_transform(weather_data[['WindGustDir']])
weather_data['WindDir9am'] = LB.fit_transform(weather_data[['WindDir9am']])
weather_data['WindDir3pm'] = LB.fit_transform(weather_data[['WindDir3pm']])

#Here, change the Labels of our main forecasts.

from sklearn.preprocessing import LabelEncoder
LE = LabelEncoder()
weather_data['RainToday'] = LE.fit_transform(weather_data['RainToday'])
weather_data['RainTomorrow'] = LE.fit_transform(weather_data['RainTomorrow'])
# How the data is doing.
weather_data.head()
```

```
   MinTemp  MaxTemp  RainFall  Evaporation  Sunshine  WindGustDir  WindGustSpeed  WindDir9am  WindDir3pm  WindSpeed9am  ...  Humidity9am  Humidity3pm  Pressure9am  Pressure3pm
0      8.0     24.3       0.0          3.4       6.3            0           30.0           0           0           6.0  ...           68           29       1019.7       1015.0
1     14.0     26.9       3.6          4.4       9.7            0           39.0           1           0           4.0  ...           80           36       1012.4       1008.4
2     13.7     23.4       3.6          5.8       3.3            0           85.0           0           0           6.0  ...           82           69       1009.5       1007.2
3     13.3     15.5      39.8          7.2       9.1            0           54.0           0           0          30.0  ...           62           56       1005.5       1007.0
4      7.6     16.1       2.8          5.6      10.6            0           50.0           0           0          20.0  ...           68           49       1018.3       1018.5
5 rows × 21 columns
```

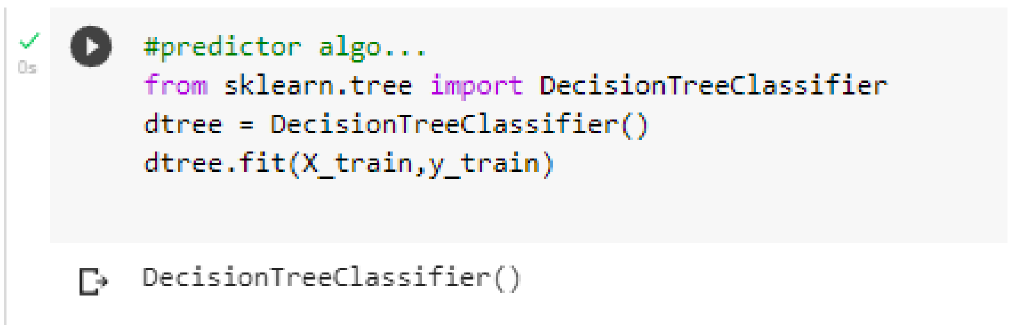
Training the Model

Separate the training and test variables.



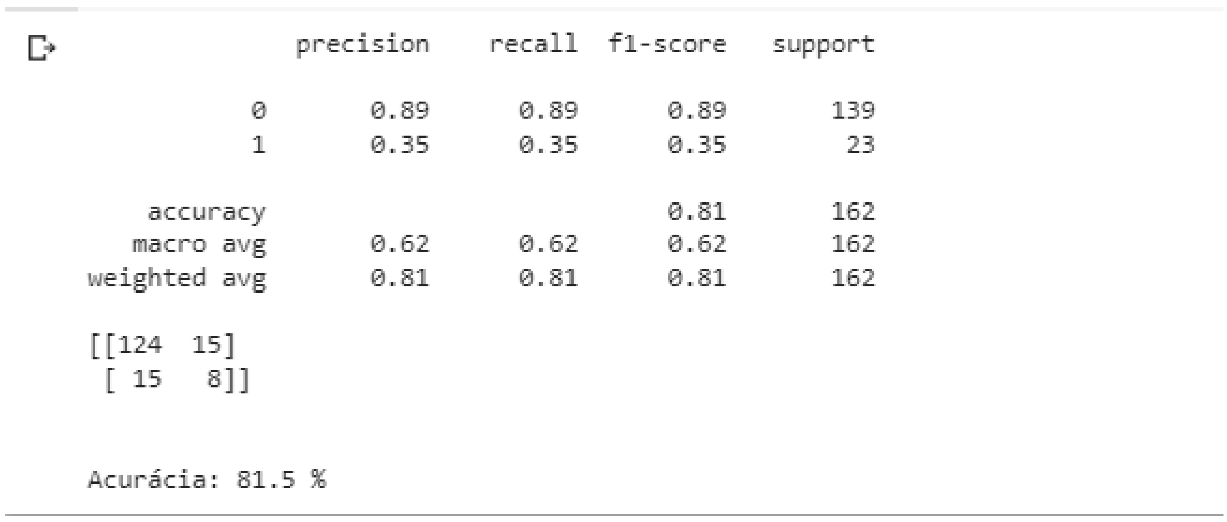
```
from sklearn.model_selection import train_test_split
X = weather_data.drop('RainTomorrow',axis=1).values
y = weather_data['RainTomorrow'].values
# standardize  information, as they are on very different scales.
from sklearn.preprocessing import MinMaxScaler
minmax = MinMaxScaler()
X = minmax.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.44, random_state=104)
```

Decision Tree Classifier



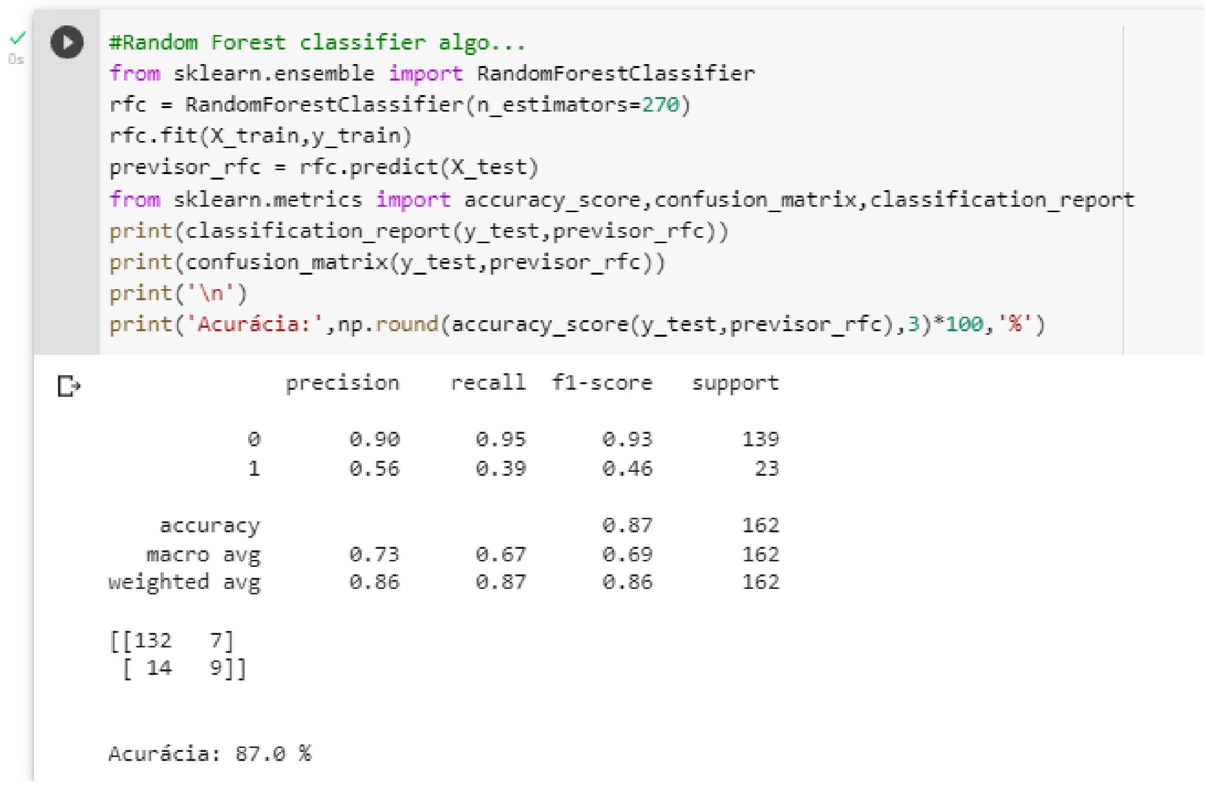
```
#predictor algo...
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```



```
              precision    recall  f1-score   support

           0       0.89      0.89      0.89       139
           1       0.35      0.35      0.35        23

    accuracy                           0.81       162
   macro avg       0.62      0.62      0.62       162
weighted avg       0.81      0.81      0.81       162

[[124  15]
 [ 15   8]]


Acurácia: 81.5 %
```

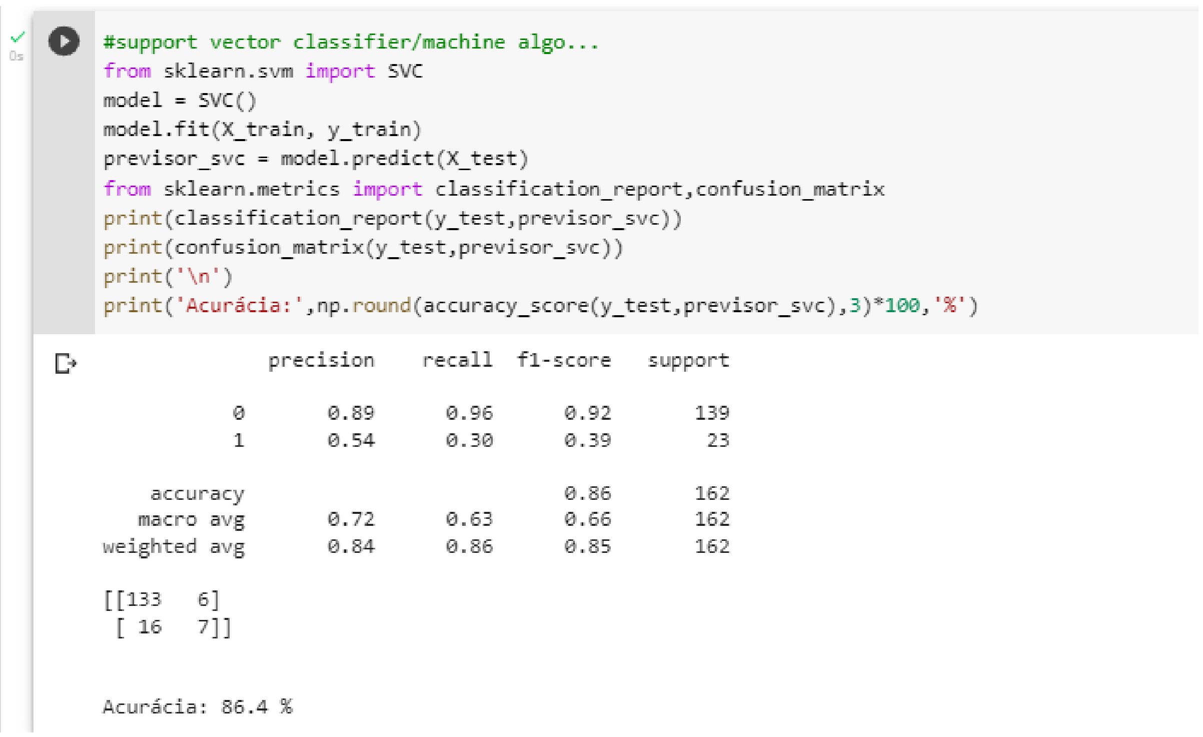
Random Forest Classifier



```
#Random Forest classifier algo...
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=270)
rfc.fit(X_train,y_train)
previsor_rfc = rfc.predict(X_test)
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
print(classification_report(y_test,previsor_rfc))
print(confusion_matrix(y_test,previsor_rfc))
print('\n')
print('Acurácia:',np.round(accuracy_score(y_test,previsor_rfc),3)*100,'%')
```

```
              precision    recall  f1-score   support

           0       0.90      0.95      0.93       139
           1       0.56      0.39      0.46        23

    accuracy                           0.87       162
   macro avg       0.73      0.67      0.69       162
weighted avg       0.86      0.87      0.86       162

[[132   7]
 [ 14   9]]


Acurácia: 87.0 %
```

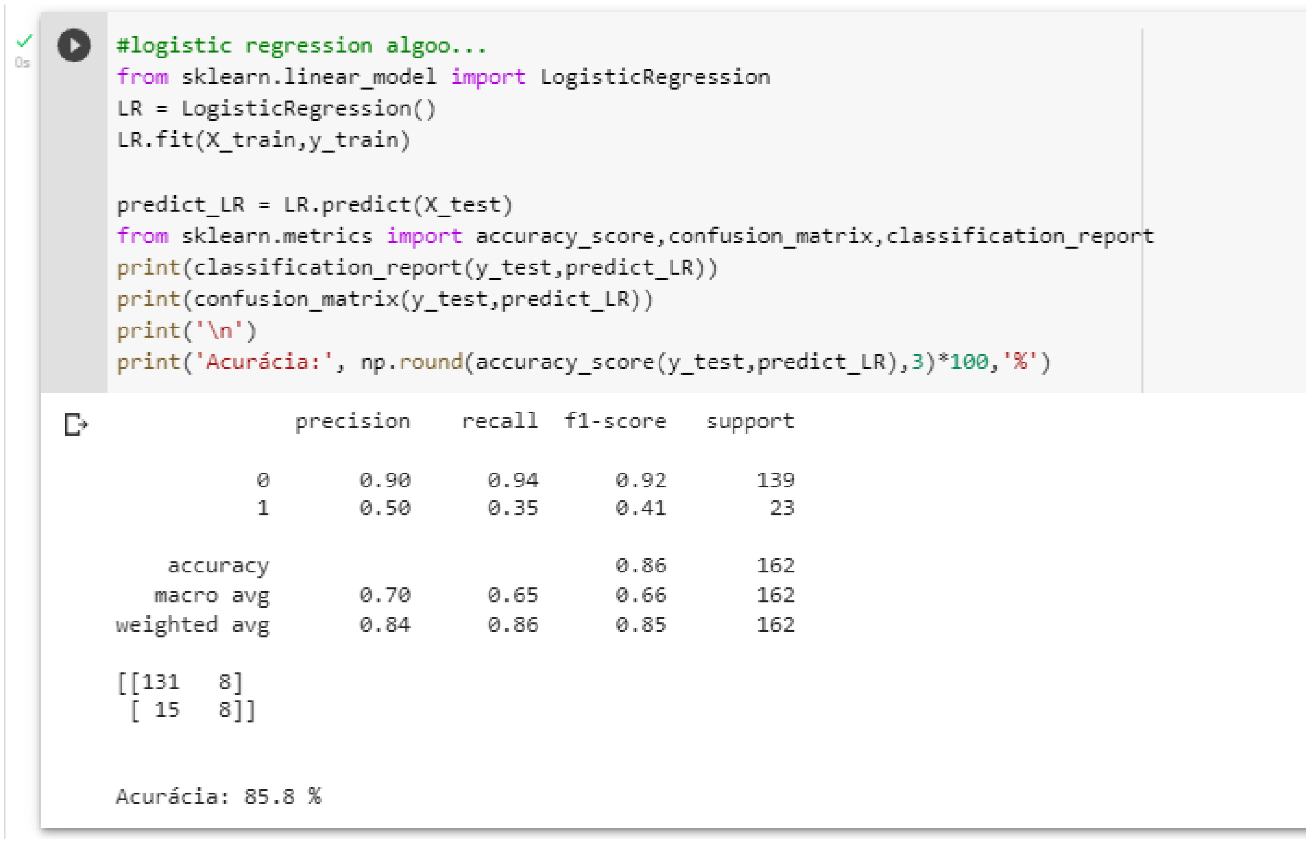
Support Vector Classifier



```
#support vector classifier/machine algo...
from sklearn.svm import SVC
model = SVC()
model.fit(X_train, y_train)
previsor_svc = model.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,previsor_svc))
print(confusion_matrix(y_test,previsor_svc))
print('\n')
print('Acurácia:',np.round(accuracy_score(y_test,previsor_svc),3)*100,'%')
```

```
              precision    recall  f1-score   support

           0       0.89      0.96      0.92       139
           1       0.54      0.30      0.39        23

    accuracy                           0.86       162
   macro avg       0.72      0.63      0.66       162
weighted avg       0.84      0.86      0.85       162

[[133   6]
 [ 16   7]]


Acurácia: 86.4 %
```

Logistic Regression



```
#logistic regression algoo...
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR.fit(X_train,y_train)

predict_LR = LR.predict(X_test)
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
print(classification_report(y_test,predict_LR))
print(confusion_matrix(y_test,predict_LR))
print('\n')
print('Acurácia:', np.round(accuracy_score(y_test,predict_LR),3)*100,'%')
```

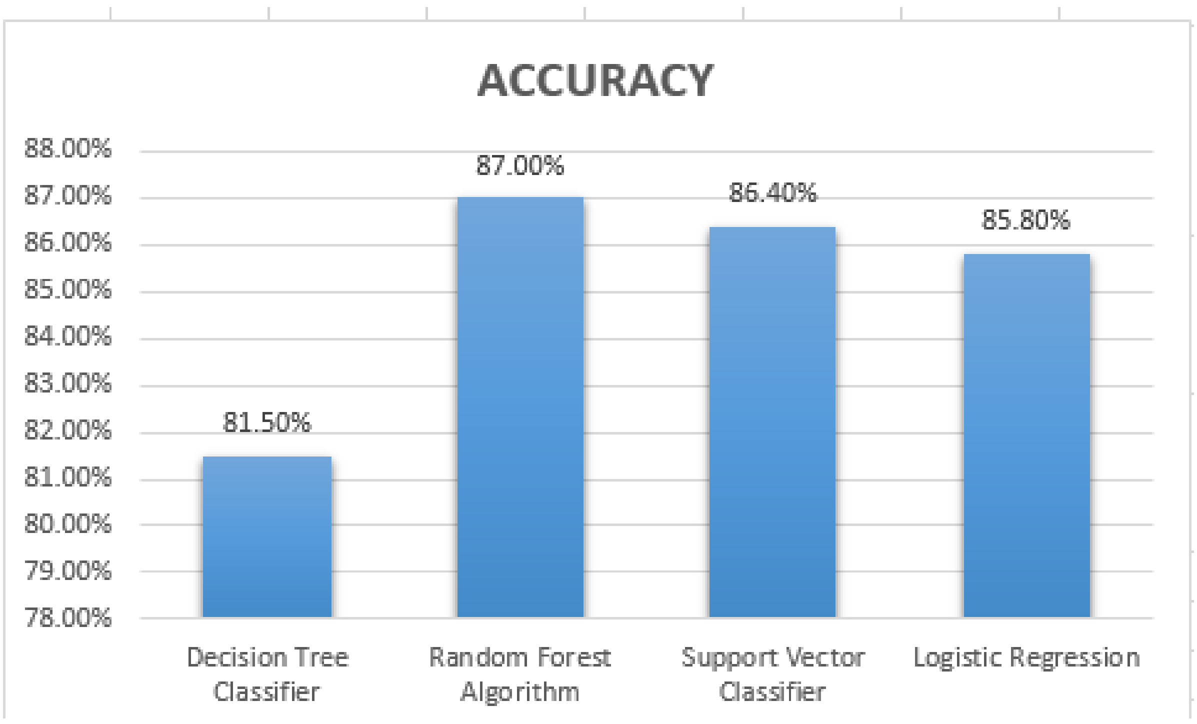
```
              precision    recall  f1-score   support

           0       0.90      0.94      0.92       139
           1       0.50      0.35      0.41        23

    accuracy                           0.86       162
   macro avg       0.70      0.65      0.66       162
weighted avg       0.84      0.86      0.85       162

[[131   8]
 [ 15   8]]


Acurácia: 85.8 %
```

ACCURACY

The accuracy of the algorithm's is listed in the table

ALGORITHM	ACCURACY
Decision Tree Classifier	81.5%
Random Forest Algorithm	87.0%
Support Vector Classifier	86.4%
Logistic Regression	85.8%

The accuracy of the algorithm's graph below



CONCLUSION

In this Project, Presented a technology to utilize machine learning techniques to provide weather forecasts. Machine learning technology can provide intelligent models, which are much simpler than traditional physical models. The confusion matrix results show that these machine learning models can predict weather features accurately enough to compete with traditional models. Weather data is considered with different attributes for weather forecasting. The weather forecasting experiment was carried out to analyse the performance of different machine learning techniques. Trained Four different models on this data Support vector classifier, Decision tree classifier, linear regression and Random forest. Then used these models to predict weather and calculated root mean square error from the actual temperature. The results achieved the accuracy of Decision Tree Classifier 81.5%, Random forest 87.0%, Support Vector Classifier 86.4%, logistic regression85.8%.. From observation of this project, Found out that time series using Random forest algorithm is a better method for weather forecasting.

REFERENCE

1) Ron Holmes, “Using a decision tree and neural net to identify severe weather radar characteristics”

2) Machine Learning Applied to Weather Forecasting - Mark A. Holmstrom, Dylan Liu-Environmental Science-2016-Arne Sund. 2015. Using Amazon Machine Learning to Predict the weather.

3) Prediction Using Machine Learning-Abhishek Patel, P. Singh, Shivam Tandon -Computer Science, Environmental Science-2020

4) Manojit Chattopadhyay, Surajit Chattopadhyay, “Elucidating the role of topological pattern discovery and support vector machine in generating predictive models for Indian summer monsoon rainfall”, Theoretical and Applied Climatology, pp. 1-12, July 2015, DOI: 10.1007/s00704-015-1544

5) Roberto Buizza, et.al, “A Comparison of the ECMWF, MSC, and NCEP global ensemble prediction systems”, Monthly Weather Review, vol. 133, No. 5, pp. 1076-1097, 2005, doi: 10.1175/MWR2905.1.

6) SandeepPattnaik, “Weather forecasting in India: Recent developments,” Mausam, vol. 70, no. 3, pp. 453-464, 2019.

7) G.Santamaría-Bon, et al., “Wind speed forecasting for wind farms: A method based on support vector regression,” Renewable Energy, vol. 85, pp. 790-809, 2015, doi: 10.1016/j.renene.2015.07.004.

8) Chujie Tian, et al., “A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network,” Energies, vol. 11, no. 12, pp. 3493, 2018, doi: 10.3390/en11123493.

9) Atul Kulkarni and Debajyoti Mukhopadhyay, “Internet of Things Based Weather Forecast Monitoring System,” Indonesian Journal of Electrical Engineering and Computer Science (IJEECS), vol. 9, no. 3, pp. 555-557, March 2018, doi: 10.11591/ijeecs.v9.i3.pp555-557