School of Information Technology & Engineering

BIG DATA ANALYSIS

ITE2013

PROJECT

RAIN PREDICTION

by

Om Purohit (17BIT0368)

Jash Shah (17BIT0337)

Sawar Pratap Singh(17BIT0270)

Under the guidance of

Prof. Kathiravan

ABSTRACT:

The biggest problem we face nowadays is not knowing whether it is going to rain or not. Imagine planning for something big, and all your plan gets cancelled just because it starts raining. Sounds pretty irritating, doesn't it? Well, now we have a solution to this problem and we call it the **Rain Predictor**. Rain Predictor is a software which will help us predict whether it is going to rain on a particular day or not. Rain Prediction will be done on the basis of different factors which affect the rain conditions, such as temperature, evaporation, humidity, wind speed, etc. On the basis of the available dataset with around 400 records of past data and the weather factors of a day, we can predict whether it is going to rain on that day or not. We are going to do the prediction of rain on the basis of different factors that are responsible for affecting the rain conditions.

LITERATURE SURVEY

Sno	Title	Dataset	Brief Desciption	Parameters	Advantages of the model	Limitations of the model
1	**Prediction of Rain Attenuation Statistics from Measured Rain Rate Statistics using Synthetic**	Rain rate data are taken from ITU-R data bank for different tropical and temperate locations to show the applicability of	Prediction of signal attenuation due to rain are important in the conception of microwave and millimetre wave communication	**Site location, Latitude, Longitude, Elevation, Frequency**	The present study shows how SST can be successfully used to convert measured rain rate statistics	**Normallya 2.2648 accuracy is a good accuracy but since this is managing realtime disasters we require**

	Storm Technique for Micro and Millimeter wave Communication System	the SST to predict rain attenuation statistics from rain rate statistics. Data sets from Kolkata,Hongkong and Singapore(in tropical region), Spino d' Adda and Walthum(two locations in temperate region) have been used in this paper	system design. Since it is not feasible to use large fade margins, such systems will implement dynamic fade countermeasure s, FCMs, to react to fading effects		into attenuation statistics at Ku and K band by properly selecting the time gap between two rain rate samples	an almost perfect model
2	Rain rate and rain attenuation prediction with experimental rain attenuation efforts in south-western Nigeria	This data is available at the Kitami Institute of Technology databank	Rain induced attenuation is a prominent loss factor for communication system design in the terrestrial and satellite- earth links. Its severity is more pronounced at frequencies above 10GHz [1]	The prominent ITU - R rain rate model as detailed in [4] is based on the use of meteorological parameters available from ITU's 3M Group website. The Kitami rain rate distribution model employs two regional	The ITU and RH models show good performance at low rain rates while Kitami model shows the worst result for the location. Fig . 3 shows the predicted rain attenuation at 12.736	The model works at certain frequency only such as 31.4 GHZ

				climatic parameters ;	GHz, 12.522 GHz,	

				the thunderstorm ratio and average annual precipitation and the 290 datasets for 30 countries (including the tropics) as proposed by Ito and Hosoya	12.722 GHz and 12.245 GHz for different percentages of an average year.	
3	Novel integration-time conversion of rain-rate statistics for rain attenuation prediction models	A disdrometer has been used for rain accumulation measurements Thirty rain events during 2011-2012 are considered``	Rain attenuation prediction model is important for both satellite and terrestrial communications.	The instability parameters are estimated from radiometric data to point the development of atmospheric	The nowcasting technique is, therefore, able to predict both rain occurrence and rain accumulation.	We get results at 22.24, 23.8, 26.4 and 31GHz but only optimal and consiferable is at 31.4Ghz
4	A model of rain attenuation in Ka band based on the Wiener prediction	DAH is a prediction model, which is proposed by Allnutt, Dissanayake and Haidara	In this paper a new rain attenuation model based on Wiener prediction is established after analyzing the DAH model in Ka	Because of more parameters , more complex process of calculation	In this paper, a new rain attenuation model is introduced based on	B5esides, this new model of 3th order has only 3 parameters, and

		after analyzing the data that come from series of experiments based on INTEL SAT satellite system,	band.	and the need of renewing all the parameters	the analysis of DAH model. Simulation results show that this new model can achieve the same effect with DAH model	there are no close relationship between the parameters and frequency.
5	A New Rain Attenuation Prediction Model for the Earth-Space Links	Based on the measurement data by Meteorological radar, a rain attenuation prediction model was	Earth-space communication systems are now utilizing the *Ku*- and *Ka*-frequency	Twelve parameters sets, one for each month of the year, are available. The model	A new rain attenuation prediction model for the earth-space links is	ut also over various ranges of latitudes, frequencies, and

6	Rain rate and rain attenuation prediction with experimental rain attenuation efforts south-western Nigeria	This data is available at the Kitami Institute of Technology databank.	Rain induced attenuation is a prominent loss factor for communication system design in the terrestrial and satelliteearth links. Its severity is more pronounced at frequencies above 10GHz	The prominent ITU - R rain rate model as detailed in [4] is based on the use of meteorological parameters available from ITU's 3M Group website. The Kitami rain rate distribution model employs two regional climatic parameters; the thunderstorm ratio and average annual precipitation and the 290 datasets for 30 countries (including the tropics) as proposed by Ito and Hosoya	The ITU and RH models show good performance at low rain rates while Kitami model shows the worst result for the location. Fig . 3 shows the predicted rain attenuation at 12.736 GHz, 12.522 GHz, 12.722 GHz and 12.245 GHz for different percentages of an average year.	The works mode certain l at frequency only such as 31.4 GHZ
7	Validation the Applicability of	Tropical Rainfall	The spatial rainfall is one of the key	The model parameters are	The results show	the similar

	Satellite Based Rainfall Data for Runoff Simulation and Water Balance Analysis	data	inputs for the distributed hydrological model, and their precisions heavily affect the accuracy of streamflow predictions from a hydrological model. Satellitebased precipitation products are expected to offer an alternative to ground based rainfall estimates in the present and the foreseeable future	optimized through a trial and error method against observed daily discharge at the Meigang station and the Nash-Sutcliffe efficiency (Ens), correlation coefficient (R2) and the relative runoff depth error (DE) were used for evaluate the model performance.	that the WATLAC (Water Flow Model for Lake Catchment) model using conventional rain gauge data produces an overall good fit, but the results for TRMM rainfall data are discontented	water balance analysis results are received, the different rainfall data source have an trivial effect on the components of water budget.
8	The SC EXCELL model for the prediction of monthly rain attenuation statistics	Preliminary tests are carried out to validate this assumption against the extensive set of rain attenuation data collected at the experimental station of Spino d'Adda during the Italsat propagation experiment	presents the prediction of monthly rain attenuation statistics by means of the Stratiform/ Convect ive (SC) EXCELL model. The assumption put forth in this contribution is that, although developed for the prediction of yearly statistics, thanks to its	probability levels equal to or higher than 5×10−3% have been considered in the tests so as to maintain a good degree of statistical stability	knowledge of which may lead to significant benefits in the design and operation of advanced systems taking advantage of the high spatial and temporal variability of	Despite the need of additional tests, overall, the results obtained in this work show that, although originally developed for the prediction of yearly

		(years 1994-2000)	solid physical basis, SC EXCELL can be applied as is also to estimate monthly rain attenuation statistics, provided that suitable inputs (i.e. monthly rain rate statistics and monthly rain height) are used	in the P(A)ms.	the rainfall process (e.g. design of communication systems for Earth Observation missions or of broadcast systems on the basis of worst month rain attenuation statistics	statistics, thanks to its solid physical basis, SC EXCELL can be used as is with satisfactory accuracy also for the prediction of P(A)ms
9	Rain Prediction Using Radiometric	A disdrometer has been	Nowcasting of intense rain is important in	The instability parameters are	The nowcasting technique is,	We get results at 22.24, 23.8, 26.4 and 31GHz but
10	Observations at a Tropical Location	used for rain accumulation measurements Thirty rain events during 2011-2012 are considered	various fields of life. In this paper, radiometric brightness temperature measurements at Ka and V bands are utilized to predict rain event associated with impending convective processes.	estimated from radiometric data to point the development of atmospheric instability and an estimation of liquid water content is made from brightness temperature at 31.4 GHz, prior to rain events	therefore, able to predict both rain occurrence and rain accumulation. The model, when validated, gives a reasonable prediction efficiency of around 80%.	only optimal and consiferable is at 31.4Ghz
11	Analysis of the Synthet	relies on rain rate time series and rain	aims to investigate the utilization of the Synthetic Storm	model requires the speed (m/s) of	investigate the utilization of the Synthetic	generate rain attenuation time series by relying on

	ic Storm Technique Using Rain Height Models to Predict Rain Attenuation in Tropical Regions	attenuation time series measured at University Science Malaysia (USM) campus	Technique (SST) to convert rain rate time series to rain attenuation time series using the ITU-R P.839, Stutz man and Bryant rain height models	rain cells, the length of link path between satellite and base station (km) and the rain rate time series which was measured at the desired site. The model consists of two layers, namely, the rain layer (layer A) and the melting layer (layer B)	Storm Technique (SST) to convert rain rate time series to rain attenuation time series	rain rate data measured for one month in USM campus

12	Flood Prediction Using Multi-Layer Artificial Neural Network in Monitoring System with Rain Gauge, Water Level, Soil Moisture Sensors	data it gathered from sensors integrated in a realtime monitoring system.	Flood is one of the most destructive natural phenomena that happens on most part of the world. Notably in the Philippines, this was a major issue as it can lead to damage of properties, damage to infrastructures or even loss	network showed a very good goodness-of-fit specifically **0.99889** for the training dataset, **0.99362** for the test data set, **0.99764** for the validation dataset and **0.99795**	The flood prediction system showed an RMSD value of **2.2648** which signifies a small overall difference between the predicted flood level and actual flood level across the whole dataset	Normallya **2.2648** accuracy is a good accuracy but since this is managing realtime disasters we require an almost perfect model

			of lives. Current systems adhere to solve issues to prevent devastating disasters caused by floods. In this study, a system is developed to predict flood level based on real-time monitoring sensors and systems.		tested in the actual setup.	
13	Validation the Applicability of Satellite Based Rainfall Data for Runoff Simulation and Water Balance Analysis	Tropical Rainfall data	The spatial rainfall is one of the key inputs for the distributed hydrological model, and their precisions heavily affect the accuracy of streamflow predictions from a hydrological model. Satellite-based precipitation products are expected to offer an alternative to	The model parameters are optimized through a trial and error method against observed daily discharge at the Meigang station and the Nash-Sutcliffe efficiency (Ens), correlation coefficient (R2) and the relative runoff depth error (DE)	The results show that the WATLAC (Water Flow Model for Lake Catchment) model using conventional rain gauge data produces an overall good fit, but the results for TRMM rainfall data are discontented	the similar water balance analysis results are received, the different rainfall data source have an trivial effect on the components of water budget.

			ground based rainfall estimates in the present and the foreseeable future	were used for evaluate the model performance.		
14	Prediction of convective rainfall using multi-technique observations	radio environment over Kolkata (22.65o N, 88.45o E) using a multi frequency profiler radiometer (RPGHATPRO) and an electric field monitor. The multi-frequency microwave radiometer measures brightness temperatures at 14 frequencies at two frequency bands	Nowcasting of rain and especially its amount is very important view of ad verse effects caused by rainfall	monitoring of brightness temperature at 31.4 GHz, the frequency having strong absorption due to liquid water, can give an estimate of rain accumulation with a prediction efficiency of ~ 80 % and a lead time of 75 minutes.	In this study, an effort has been made to predict the quantity of precipitation using multi-technique observations at a tropical location. It is seen that both the brightness temperature at 31.4 GHz and the atmospheric electric field show definite changes before convective rain	The model works at certain frequency only such as 31.4 GHZ
15	Rain Prediction Using Radiometric Observations at a Tropical Location	A disdrometer has been used for rain accumulation measurements Thirty rain	Nowcasting of intense rain is important in various fields of life. In this paper, radiometric	The instability parameters are estimated from radiometric data to point	The nowcasting technique is, therefore, able to predict both rain occurrence	We get results at 22.24, 23.8, 26.4 and 31GHz but only optimal and consiferable is at

		events during 2011-2012 are considered	brightness temperature measurements at Ka and V bands are utilized to predict rain event associated with impending convective processes.	the development of atmospheric instability and an estimation of liquid water content is made from brightness temperature at 31.4 GHz, prior to rain events	and rain accumulation. The model, when validated, gives a reasonable prediction efficiency of around 80%.	31.4Ghz
16	Modelling Rain Rate by Means of Arrival Processes	This is the ITU-R, standard for predicting rain induced attenuation For one, the availability of rain intensity records adapted to the ITU-R requirements	address in this paper current work toward the generation of synthetic time series of rain rate, R, in mm/h with an integration time of 1 min.	Twelve parameters sets, one for each month of the year, are available. The model assumes three states: (1) intense rainy state, (2) wet state, and (3) dry state (little or no rain)	It is expected that through the study of arrivals: first of rain gauge tips and, possibly, drop arrivals, a more faithful reproduction of rain induced effects on radio links could be achieved.	ITU-R requirements is very limited given that the main users of rain information (weather prediction, water management, etc.) do not require such a short integration time and data with longer accumulation periods, even exceeding 1 hour, are commonly

						found.

BIG DATA LIFE CYCLE

PHASE 1: Discovery

- Domain knowledge

  Our project is to predict that on the given weather conditions what is the possibility of rain to happen. Rain is the collection of water droplets that falls from the clouds. Cloud takes water in form vapours from oceans, trees, lakes etc. The vapours coming from these sources are dependent on different factors such as, if temperature is high then the possibility of occurrence rain is more. Similarly, it depends on evaporitic pressure and sunshine too. Humidity, wind direction and wind speed are also playing a very important role. If humidity is high then chances of rain are much more. Wind direction and wind speed actually tells us that when and where the rain is going to happen.

- Resources

  The data required can be taken from different resources.

  The given data can be taken from two resources –

  1) **From Sensors**

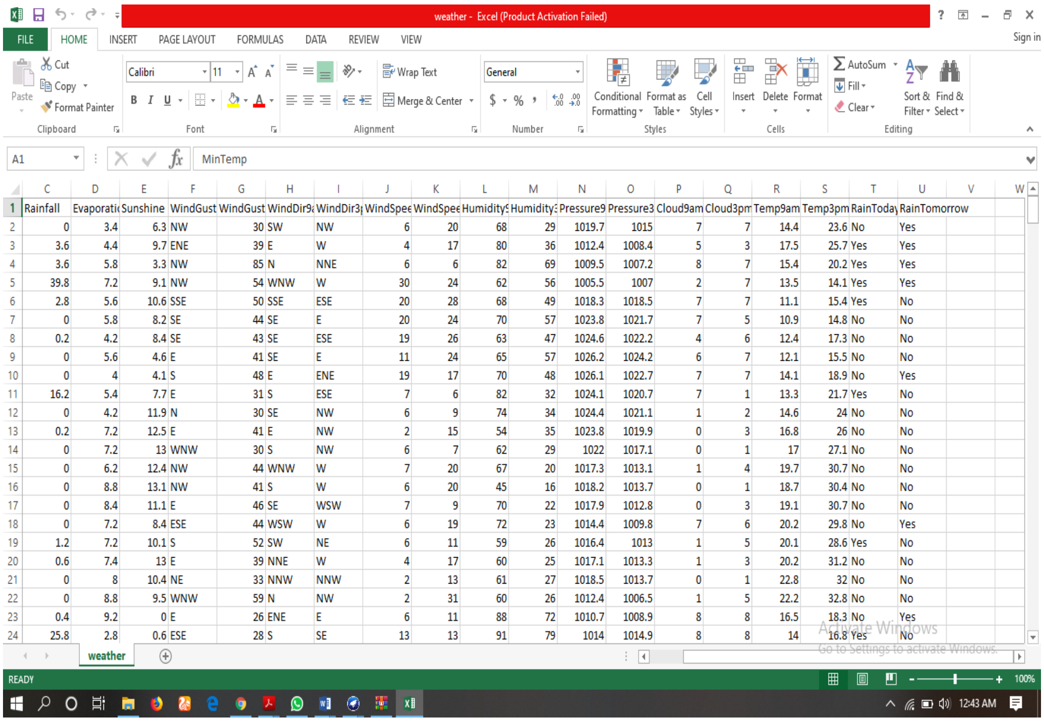
     The humidity, temperature, evaporitic vapours are the data which can be recorded with the help of different types of sensors.

  2) **From satellite**

     The data like wind speed, wind direction and clouds can be obtained from satellite.

  Link- https://www.kaggle.com/arpina/weather

- Data



1	Rainfall	Evaporati	Sunshine	WindGust	WindGust	WindDir9a	WindDir3p	WindSpee	WindSpee	Humidity9	Humidity3	Pressure9	Pressure3	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
2	0	3.4	6.3	NW	30	SW	NW	6	20	68	29	1019.7	1015	7	7	14.4	23.6	No	Yes
3	3.6	4.4	9.7	ENE	39	E	W	4	17	80	36	1012.4	1008.4	5	3	17.5	25.7	Yes	Yes
4	3.6	5.8	3.3	NW	85	N	NNE	6	6	82	69	1009.5	1007.2	8	7	15.4	20.2	Yes	Yes
5	39.8	7.2	9.1	NW	54	WNW	W	30	24	62	56	1005.5	1007	2	7	13.5	14.1	Yes	Yes
6	2.8	5.6	10.6	SSE	50	SSE	ESE	20	28	68	49	1018.3	1018.5	7	7	11.1	15.4	Yes	No
7	0	5.8	8.2	SE	44	SE	E	20	24	70	57	1023.8	1021.7	7	5	10.9	14.8	No	No
8	0.2	4.2	8.4	SE	43	SE	ESE	19	26	63	47	1024.6	1022.2	4	6	12.4	17.3	No	No
9	0	5.6	4.6	E	41	SE	E	11	24	65	57	1026.2	1024.2	6	7	12.1	15.5	No	No
10	0	4	4.1	S	48	E	ENE	19	17	70	48	1026.1	1022.7	7	7	14.1	18.9	No	Yes
11	16.2	5.4	7.7	E	31	S	ESE	7	6	82	32	1024.1	1020.7	7	1	13.3	21.7	Yes	No
12	0	4.2	11.9	N	30	SE	NW	6	9	74	34	1024.4	1021.1	1	2	14.6	24	No	No
13	0.2	7.2	12.5	E	41	E	NW	2	15	54	35	1023.8	1019.9	0	3	16.8	26	No	No
14	0	7.2	13	WNW	30	S	NW	6	7	62	29	1022	1017.1	0	1	17	27.1	No	No
15	0	6.2	12.4	NW	44	WNW	W	7	20	67	20	1017.3	1013.1	1	4	19.7	30.7	No	No
16	0	8.8	13.1	NW	41	S	W	6	20	45	16	1018.2	1013.7	0	1	18.7	30.4	No	No
17	0	8.4	11.1	E	46	SE	WSW	7	9	70	22	1017.9	1012.8	0	3	19.1	30.7	No	No
18	0	7.2	8.4	ESE	44	WSW	W	6	19	72	23	1014.4	1009.8	7	6	20.2	29.8	No	Yes
19	1.2	7.2	10.1	S	52	SW	NE	6	11	59	26	1016.4	1013	1	5	20.1	28.6	Yes	No
20	0.6	7.4	13	E	39	NNE	W	4	17	60	25	1017.1	1013.3	1	3	20.2	31.2	No	No
21	0	8	10.4	NE	33	NNW	NNW	2	13	61	27	1018.5	1013.7	0	1	22.8	32	No	No
22	0	8.8	9.5	WNW	59	N	NW	2	31	60	26	1012.4	1006.5	1	5	22.2	32.8	No	No
23	0.4	9.2	0	E	26	ENE	E	6	11	88	72	1010.7	1008.9	8	8	16.5	18.3	No	Yes
24	25.8	2.8	0.6	ESE	28	S	SE	13	13	91	79	1014	1014.9	8	8	14	16.8	Yes	No

- Stakeholder & User
  1. Private weather services- they are the weather data providers. A single weather service cannot succeed on basis of its collected data, but requires contribution from different services. The system should accept this historical weather data which shall be helpful for developing patterns for rain prediction.
  2. Weather researchers- End users of the system. These stakeholders would be studying the predictions provided by our system.
  3. Requirement Engineers- This stakeholder works with customers and other stakeholder to translate needs into requirement. Specifies domain, categories, requirements into functional and non-functional. Refines requirements as needed.
  4. Software Architect- This stakeholder is the lead in development of prediction system. He will be responsible for the architecture of the system, guides design and implementation.
  5. Project Manager- This stakeholder is the lead in development of prediction system. The manager will be required to plan, manage, coordinate interactions and keep the team focused.
  6. Sponsors – People who will be providing funds for the project.

- Framing of problem

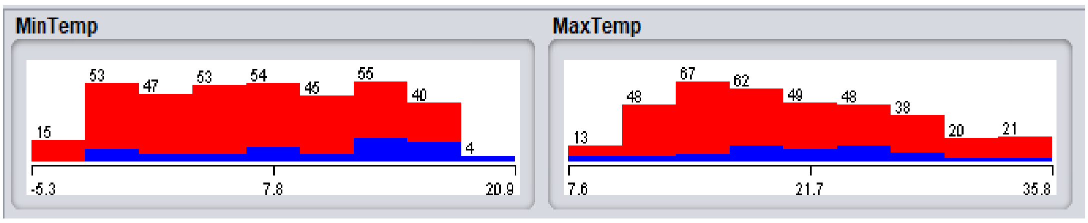
Today, the major problem that we are facing is the accurate prediction of rain. It is difficult because we are unaware of the factors affecting the rain and the extent to which these factors affect it. Rain Prediction is a complex and often challenging skill that involves observing and processing vast amounts of data. It can range from small, short lived thunderstorms only a few miles in diameter that last a couple hours to large scale rain and snow storms up to a thousand miles in diameter and lasting for days. Therefore, obtaining a suitable model can be a tedious task.

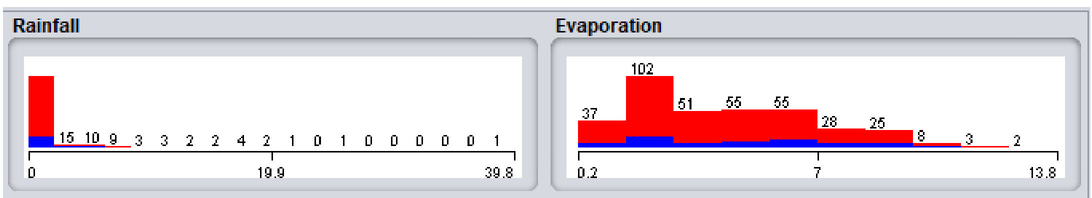
PHASE 2- DATA PREPARATION

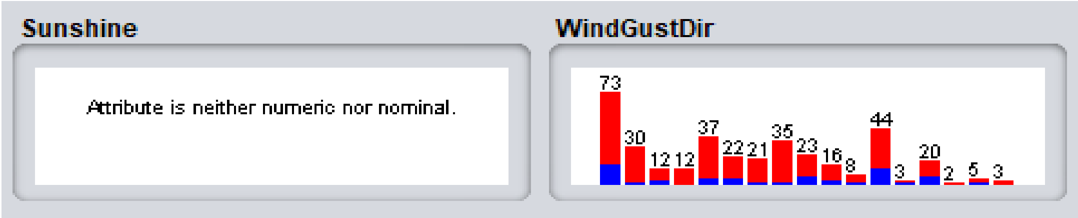
➢ Preprocessing

Data pre-processing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income:−100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), missing values, etc.

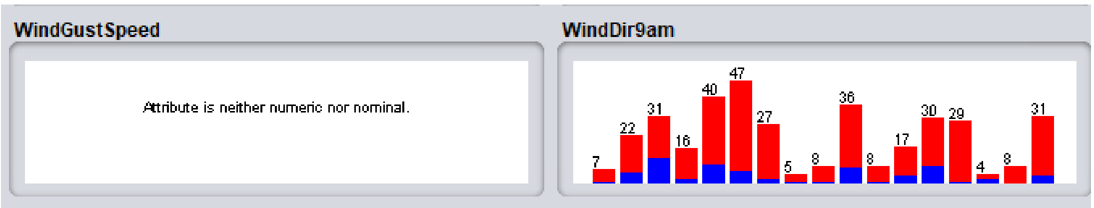
Here we have done preprocessing using weka tool. In this, we first import the dataset & then visualize all the attributes of the data one by one. Such that we come to know about the attributes that which of them attribute is neither numeric nor nominal. After identifying those attributes we removed those attribute before applying any algorithm such as decision tree, random forest, etc. So that we can get to that which algorithm will be suited best for to get the maximum percentage of correctly classified data.



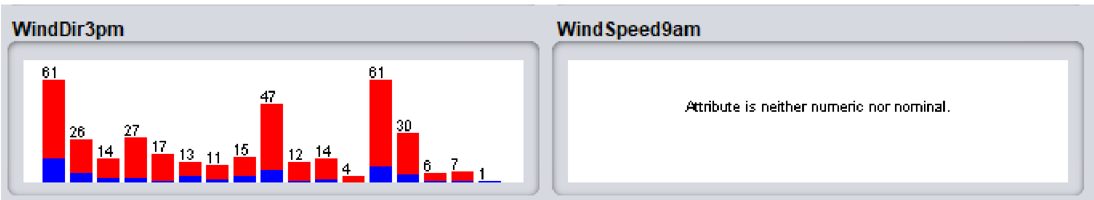




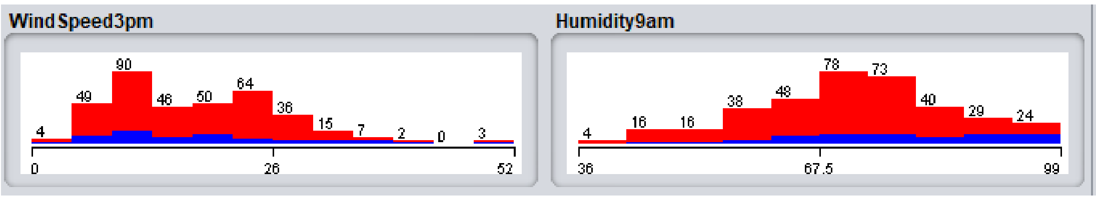
Sunshine
Attribute is neither numeric nor nominal.
WindGustDir
73 30 12 12 37 22 21 35 23 16 8 44 3 20 2 5 3



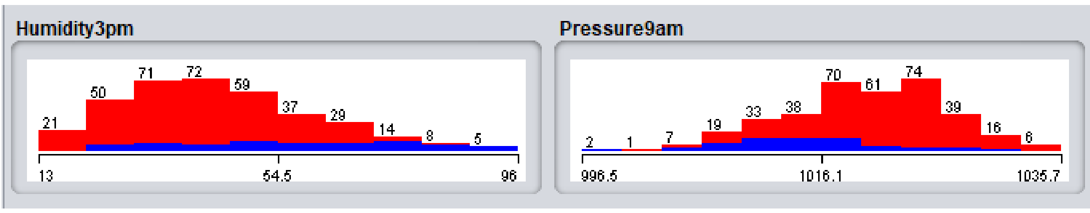
WindGustSpeed
Attribute is neither numeric nor nominal.
WindDir9am
7 22 31 16 40 47 27 5 8 36 8 17 30 29 4 8 31



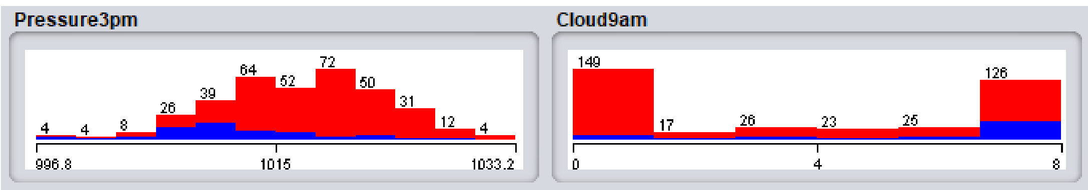
WindDir3pm
61 26 14 27 17 13 11 15 47 12 14 4 61 30 6 7 1
WindSpeed9am
Attribute is neither numeric nor nominal.



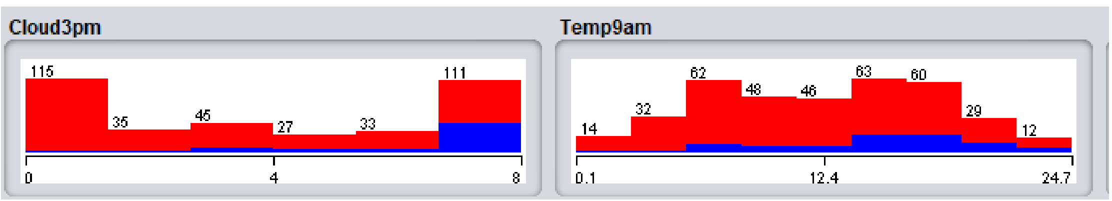
WindSpeed3pm
4 49 90 46 50 64 36 15 7 2 0 3
0 26 52
Humidity9am
4 16 16 38 48 78 73 40 29 24
36 67.5 99



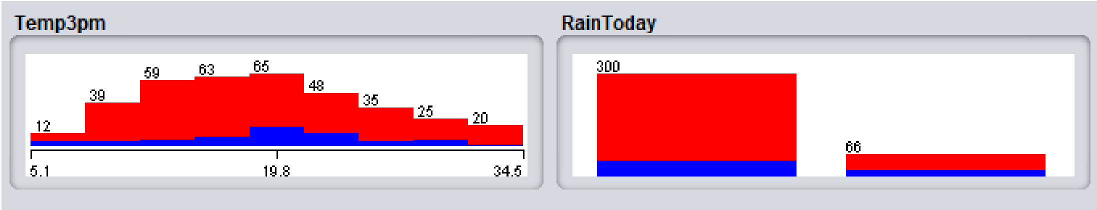
Humidity3pm
21 50 71 72 59 37 29 14 8 5
13 54.5 96
Pressure9am
2 1 7 19 33 38 70 61 74 39 16 6
996.5 1016.1 1035.7

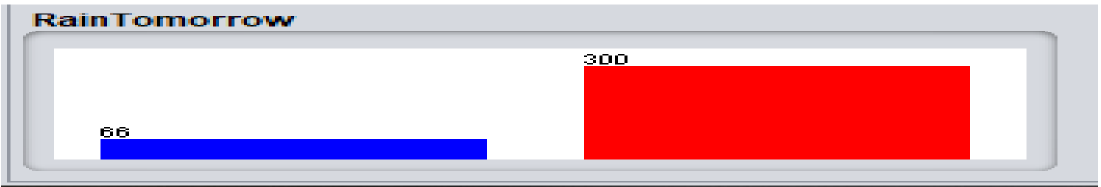


Pressure3pm
4 4 8 26 39 64 52 72 50 31 12 4
996.8 1015 1033.2
Cloud9am
149 17 26 23 25 126
0 4 8



Cloud3pm
115 35 45 27 33 111
0 4 8
Temp9am
14 32 62 48 46 63 60 29 12
0.1 12.4 24.7





- **Feature selection**

This preprocessed data tells us about the unwanted attribute present in the data. Here it shows that the attributes Sunshine, WindGustSpeed, WindSpeed9am are neither numeric nor nominal. Therefore, for doing the predictions we have to remove these attributes.

PHASE 3- MODEL PLANNING

Proposed System/Models:

RANDOM FOREST

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.[Random decision forests correct for decision trees' habit of overfitting to their training set

The first algorithm for random decision forests was created by Tin Kam Ho using the random subspace method which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler who registered "Random Forests" as a trademark (as of 2019, owned by Minitab, Inc.). The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman in order to construct a collection of decision trees with controlled variance.

NAIVE BAYES

In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models. But they could be coupled with Kernel density estimation and achieve higher accuracy levels.

Naïve Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the text retrieval community in the early 1960s,and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (document categorization)(such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines.It also finds application in automatic medical diagnosis.

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression,718 which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

K MEANS CLUSTERING

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. It is popular for cluster analysis in data mining. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes

Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

DECISION TREE

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

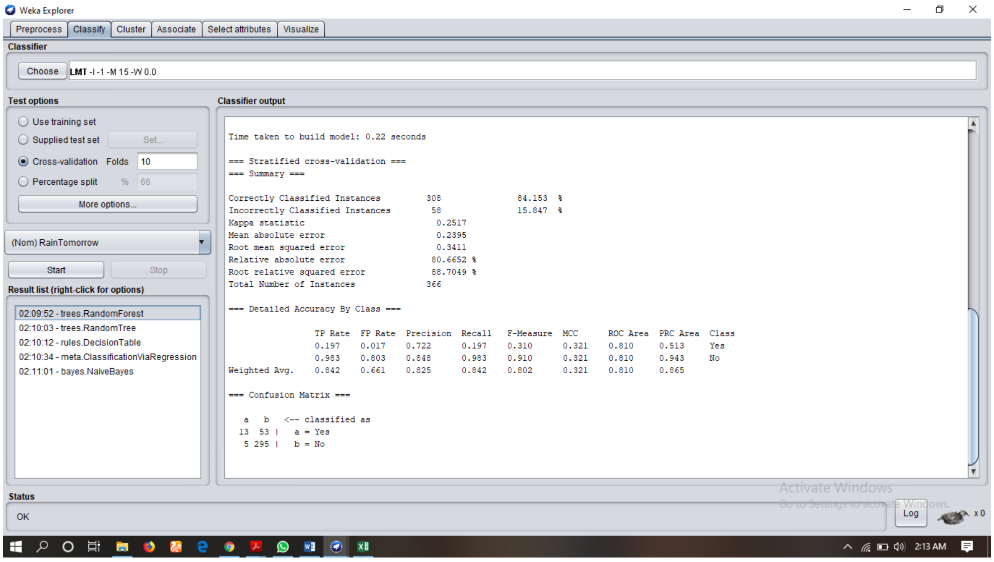
Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

Different classification algorithms are tested for building the suitable model.

The following are-

- **Random Forest**

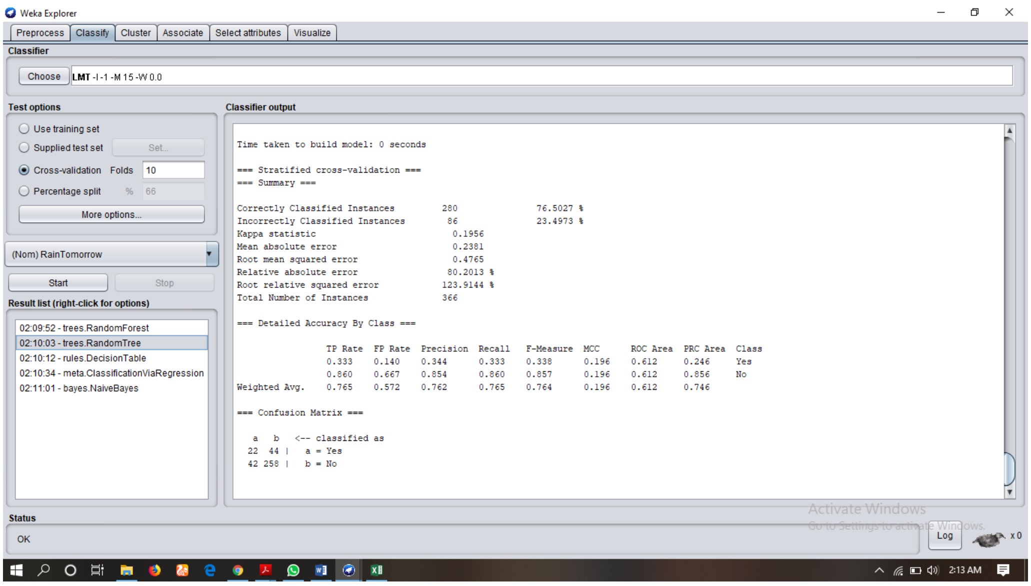
Correctly Classified Instances	308	84.153 %
Incorrectly Classified Instances	58	15.847 %



➢ Random Tree

Correctly Classified Instances 280 76.5027 %

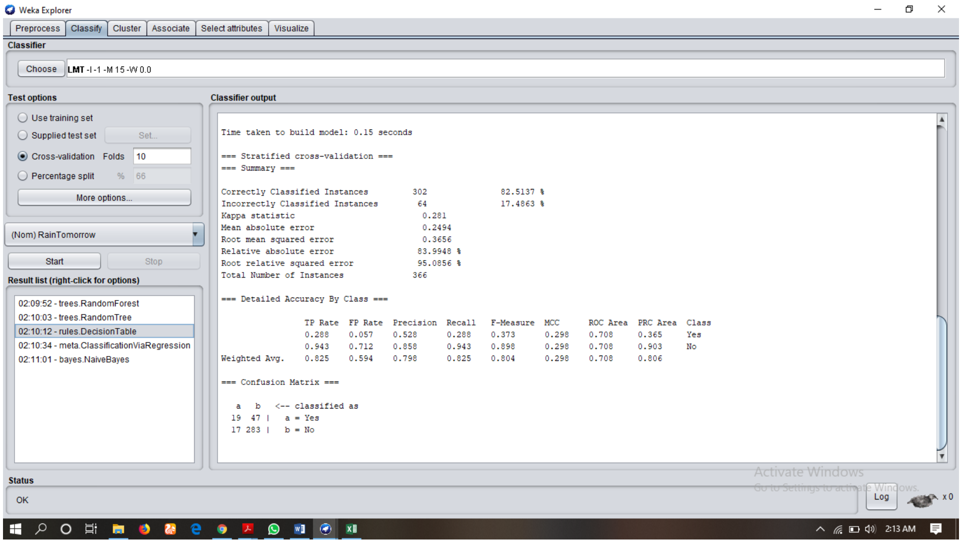
Incorrectly Classified Instances 86 23.4973 %



➢ Decision Table

Correctly Classified Instances 302 82.5137 %

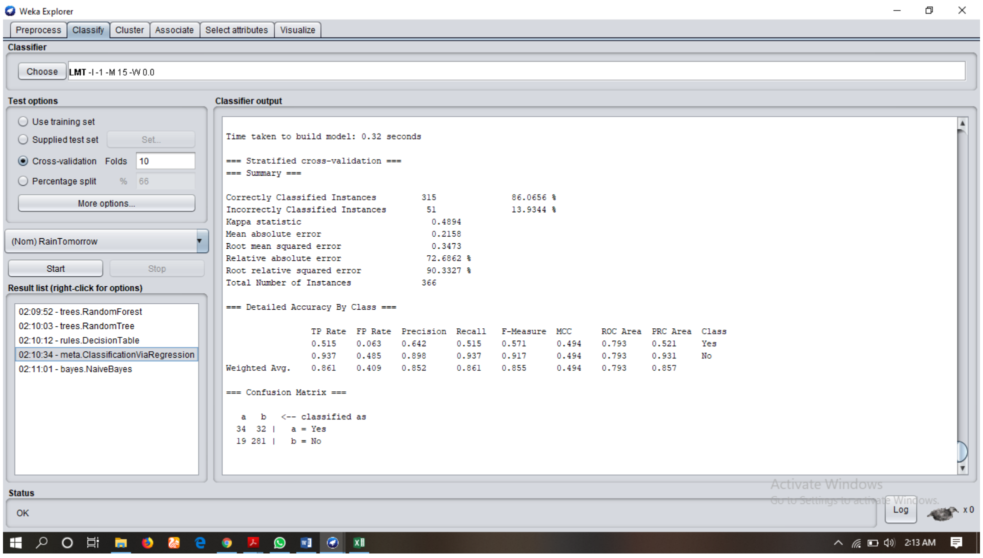
Incorrectly Classified Instances 64 17.4863 %



➢ Classification via Regression

Correctly Classified Instances 315 86.0656 %

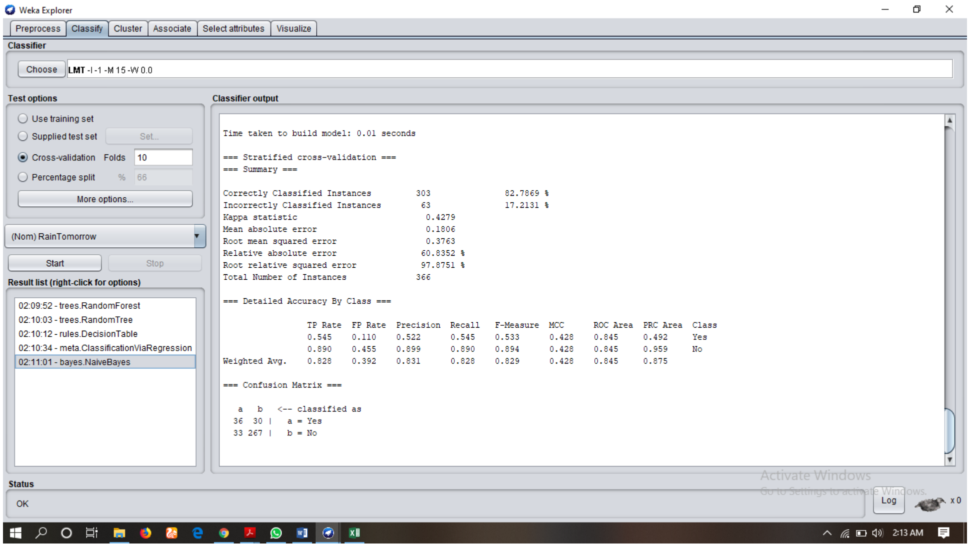
Incorrectly Classified Instances 51 13.9344 %



➢ Naïve Bayes

Correctly Classified Instances 303 82.7869 %

Incorrectly Classified Instances 63 17.2131 %



PHASE 4- MODEL BUILDING

On classifying the data by applying various algorithms like Random forest, random tree, decision table, classification via regression and naïve Bayes, accuracy percentage of regression is maximum. Thus, we could state that classification via regression is best suited as the percentage of correctly classified data is maximum in it. Next suited algorithm that can be used is Random forest.

Regression analysis consists of a set of machine learning methods that allow us to predict a continuous outcome variable (y) ,according to our dataset it will predict whether it will rain or not, based on the value of one or multiple predictor variables (x) such as temperature, humidity, wind etc. Briefly the goal of regression model is to build a mathematical equation that defines y as a function of the x variables. Next, this equation can be used to predict the outcome (y) on the basis of new values of the predictor variables (x).

We'll randomly split the data into training set (80% for building a predictive model) and test set (20% for evaluating the model).

The phases of Model Planning and Model Building overlap quite a bit, and in practice one can iterate back and forth between the two phases for a while before settling on a final model

PHASE 5- COMMUNICATION RESULTS

As the best suited model is regression model, thus for assessing the overall performance of a regression model, the most commonly known evaluation metrics include:

1. **R-squared** (R2), which is the proportion of variation in the outcome that is explained by the predictor variables. In multiple regression models, R2 corresponds to the squared correlation between the observed outcome values and the predicted values by the model. The Higher the R-squared, the better the model.
2. **Root Mean Squared Error** (RMSE), which measures the average error performed by the model in predicting the outcome for an observation. Mathematically, the RMSE is the square root of the *mean squared error (MSE)*, which is the average squared difference between the observed actual outcome values and the values predicted by the model. So, MSE = mean ((observeds - predicteds) ^ 2) and RMSE = sqrt (MSE). The lower the RMSE, the better the model.
3. **Residual Standard Error** (RSE), also known as the *model sigma*, is a variant of the RMSE adjusted for the number of predictors in the model. The lower the RSE, the better the model. In practice, the difference between RMSE and RSE is very small, particularly for large multivariate data.
4. **Mean Absolute Error** (MAE), like the RMSE, the MAE measures the prediction error. Mathematically, it is the average absolute difference between observed and predicted outcomes, MAE = mean (abs (observeds - predicteds)). MAE is less sensitive to outliers compared to RMSE.

These metrics are also used as the basis of model comparison and optimal model selection. These regression metrics are all internal measures, that is they have been computed on the same data that was used to build the regression model. They tell you how well the model fits to the data in hand, called training data set.

We check whether the model we build is a success or a failure by comparing the outcomes to our criteria.

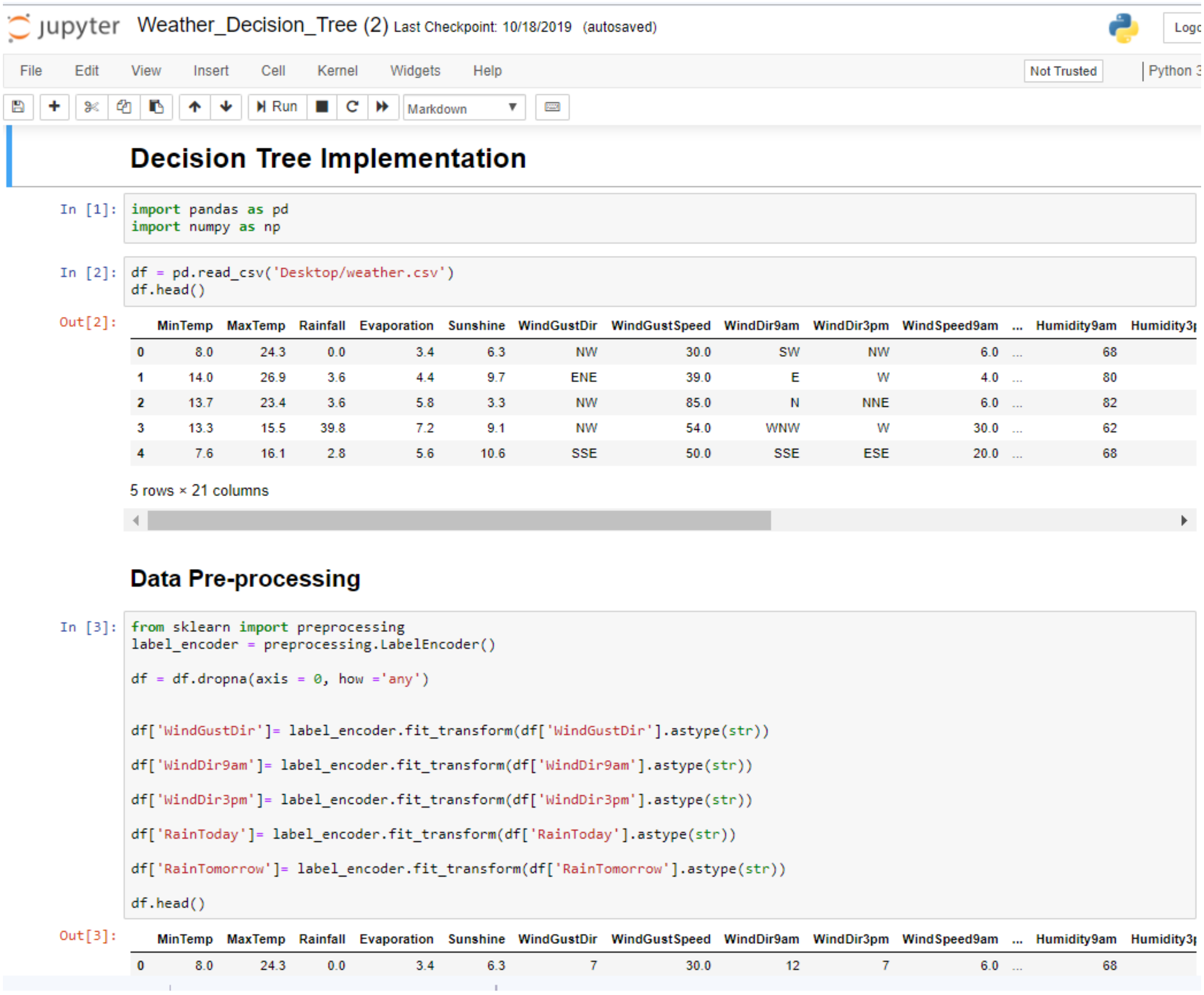
PHASE 6- OPERATIONALIZE

In this phase, we will need to assess the benefits of the work that’s been done, that is with the help of the model we will get to know the status of rain today and tomorrow, and setup a pilot so we can deploy the work in a controlled way before broadening the work to a full enterprise or ecosystem of users.
This phase can bring in a new set of team members – namely those engineers who are responsible for the production environment, who have a new set of issues and concerns. They want to ensure that running the model fits smoothly

into the production environment and the model can be integrated into downstream processes.
Assess whether the model is meeting goals and expectations, and if desired changes are actually occurring. If these outcomes are not occurring, determine if this is due to a model inaccuracy, or if its predictions are not being acted on appropriately. If needed, automate the retraining/updating of the model. In any case, we will need ongoing monitoring of model accuracy, and if accuracy degrades, we will need to retrain the model.
Final result is determined in this phase, model along with the documents are submitted and delivered.

Implementation screenshots:

Decision tree


jupyter Weather_Decision_Tree (2) Last Checkpoint: 10/18/2019 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Decision Tree Implementation

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv('Desktop/weather.csv')
df.head()
```

Out[2]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3p
0	8.0	24.3	0.0	3.4	6.3	NW	30.0	SW	NW	6.0	...	68	
1	14.0	26.9	3.6	4.4	9.7	ENE	39.0	E	W	4.0	...	80	
2	13.7	23.4	3.6	5.8	3.3	NW	85.0	N	NNE	6.0	...	82	
3	13.3	15.5	39.8	7.2	9.1	NW	54.0	WNW	W	30.0	...	62	
4	7.6	16.1	2.8	5.6	10.6	SSE	50.0	SSE	ESE	20.0	...	68	

5 rows × 21 columns

Data Pre-processing

In [3]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

df = df.dropna(axis = 0, how ='any')


df['WindGustDir']= label_encoder.fit_transform(df['WindGustDir'].astype(str))

df['WindDir9am']= label_encoder.fit_transform(df['WindDir9am'].astype(str))

df['WindDir3pm']= label_encoder.fit_transform(df['WindDir3pm'].astype(str))

df['RainToday']= label_encoder.fit_transform(df['RainToday'].astype(str))

df['RainTomorrow']= label_encoder.fit_transform(df['RainTomorrow'].astype(str))

df.head()
```

Out[3]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3p
0	8.0	24.3	0.0	3.4	6.3	7	30.0	12	7	6.0	...	68	

```
df.head()
```

Out[3]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3p
0	8.0	24.3	0.0	3.4	6.3	7	30.0	12	7	6.0	...	68	
1	14.0	26.9	3.6	4.4	9.7	1	39.0	0	13	4.0	...	80	
2	13.7	23.4	3.6	5.8	3.3	7	85.0	3	5	6.0	...	82	
3	13.3	15.5	39.8	7.2	9.1	7	54.0	14	13	30.0	...	62	
4	7.6	16.1	2.8	5.6	10.6	10	50.0	10	2	20.0	...	68	

5 rows × 21 columns

Feature Selection

In [4]:
```
X = df.iloc[:, 0:20].values
y = df.iloc[:, 20].values
```

In [5]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

Training the model

In [6]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Out[6]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [7]:
```
y_pred = classifier.predict(X_test)
```

Evaluating the model

In [8]:
```
from sklearn.metrics import classification_report,  accuracy_score

print(classification_report(y_test, y_pred))
```


Out[6]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [7]:
```
y_pred = classifier.predict(X_test)
```

Evaluating the model

In [8]:
```
from sklearn.metrics import classification_report,  accuracy_score

print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.83      0.87        66
           1       0.48      0.62      0.54        16

   micro avg       0.79      0.79      0.79        82
   macro avg       0.69      0.73      0.70        82
weighted avg       0.82      0.79      0.80        82

0.7926829268292683
```

Demo Prediction

In [9]:
```
demo=[[10,29.5,1,4,4.1,6,48,5,6,19,17,70,48,1026.1,1022.7,7,8,14.1,18.9,0],[8,29.5,1,5,4.1,6,58,5,7,19,17,70,48,1026.1,1022.7,8,
y=classifier.predict(demo)
int(round(y[1],0))
```

Out[9]: 0

In [ ]:

K-MEAN CLUSTERING

jupyter Weather_K_Means_Clustering Last Checkpoint: 10/18/2019 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Markdown

K Means Clustering Implementation

In [15]:

```
import pandas as pd
import numpy as np
```

In [20]:

```
df = pd.read_csv('weather1.csv.csv')
df.head()
```

Out[20]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3p
0	8.0	24.3	0.0	3.4	6.3	NW	30.0	SW	NW	6.0	...	68	
1	14.0	26.9	3.6	4.4	9.7	ENE	39.0	E	W	4.0	...	80	
2	13.7	23.4	3.6	5.8	3.3	NW	85.0	N	NNE	6.0	...	82	
3	13.3	15.5	39.8	7.2	9.1	NW	54.0	WNW	W	30.0	...	62	
4	7.6	16.1	2.8	5.6	10.6	SSE	50.0	SSE	ESE	20.0	...	68	

5 rows × 21 columns

Data pre-processing

In [21]:

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

df = df.dropna(axis = 0, how ='any')


df['WindGustDir']= label_encoder.fit_transform(df['WindGustDir'].astype(str))

df['WindDir9am']= label_encoder.fit_transform(df['WindDir9am'].astype(str))

df['WindDir3pm']= label_encoder.fit_transform(df['WindDir3pm'].astype(str))
```

```
df['WindGustDir']= label_encoder.fit_transform(df['WindGustDir'].astype(str))

df['WindDir9am']= label_encoder.fit_transform(df['WindDir9am'].astype(str))

df['WindDir3pm']= label_encoder.fit_transform(df['WindDir3pm'].astype(str))

df['RainToday']= label_encoder.fit_transform(df['RainToday'].astype(str))

df['RainTomorrow']= label_encoder.fit_transform(df['RainTomorrow'].astype(str))

df.head()
```

Out[21]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3p
0	8.0	24.3	0.0	3.4	6.3	7	30.0	12	7	6.0	...	68	
1	14.0	26.9	3.6	4.4	9.7	1	39.0	0	13	4.0	...	80	
2	13.7	23.4	3.6	5.8	3.3	7	85.0	3	5	6.0	...	82	
3	13.3	15.5	39.8	7.2	9.1	7	54.0	14	13	30.0	...	62	
4	7.6	16.1	2.8	5.6	10.6	10	50.0	10	2	20.0	...	68	

5 rows × 21 columns

Feature Selection

In [22]:

```
X = df.iloc[:, 0:20].values
y = df.iloc[:, 20].values
```

In [23]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

```
        n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
        random_state=None, tol=0.0001, verbose=0)
```

In [25]:
```
y_pred=km.predict(X_test)
```

Evaluating the Model

In [26]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

```
[[30 36]
 [11  5]]
              precision    recall  f1-score   support

           0       0.73      0.45      0.56        66
           1       0.12      0.31      0.18        16

    accuracy                           0.43        82
   macro avg       0.43      0.38      0.37        82
weighted avg       0.61      0.43      0.49        82

0.4268292682926829
```

Demo Prediction

In [27]:
```
demo=[[10,29.5,1,4,4.1,6,48,5,6,19,17,70,48,1026.1,1022.7,7,8,14.1,18.9,0],[8,29.5,1,5,4.1,6,58,5,7,19,17,70,48,1026.1,1022.7,8,
y=km.predict(demo)
int(round(y[1],0))
```

Out[27]: 1

In [ ]:

RANDOM FOREST

jupyter Weather_Random_Forest Last Checkpoint: 10/18/2019 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Markdown

Random Forest Implementation

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv('weather1.csv.csv')
df.head()
```

Out[2]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3
0	8.0	24.3	0.0	3.4	6.3	NW	30.0	SW	NW	6.0	...	68	
1	14.0	26.9	3.6	4.4	9.7	ENE	39.0	E	W	4.0	...	80	
2	13.7	23.4	3.6	5.8	3.3	NW	85.0	N	NNE	6.0	...	82	
3	13.3	15.5	39.8	7.2	9.1	NW	54.0	WNW	W	30.0	...	62	
4	7.6	16.1	2.8	5.6	10.6	SSE	50.0	SSE	ESE	20.0	...	68	

5 rows × 21 columns

Data Pre-processing

In [3]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

df = df.dropna(axis = 0, how ='any')


df['WindGustDir']= label_encoder.fit_transform(df['WindGustDir'].astype(str))

df['WindDir9am']= label_encoder.fit_transform(df['WindDir9am'].astype(str))

df['WindDir3pm']= label_encoder.fit_transform(df['WindDir3pm'].astype(str))

df['RainToday']= label_encoder.fit_transform(df['RainToday'].astype(str))
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3p
0	8.0	24.3	0.0	3.4	6.3	7	30.0	12	7	6.0	...	68	
1	14.0	26.9	3.6	4.4	9.7	1	39.0	0	13	4.0	...	80	
2	13.7	23.4	3.6	5.8	3.3	7	85.0	3	5	6.0	...	82	
3	13.3	15.5	39.8	7.2	9.1	7	54.0	14	13	30.0	...	62	
4	7.6	16.1	2.8	5.6	10.6	10	50.0	10	2	20.0	...	68	

5 rows × 21 columns

Feature Selection

In [4]:
```
X = df.iloc[:, 0:20].values
y = df.iloc[:, 20].values
```

In [5]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

Training the model

In [6]:
```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=20, random_state=0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

Evaluating the model

In [7]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

y_pred = regressor.predict(X_test)

Evaluating the model

In [7]: from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

```
print(classification_report(y_test, y_pred.round()))
print(accuracy_score(y_test, y_pred.round()))
```

```
              precision    recall  f1-score   support

           0       0.86      0.92      0.89        66
           1       0.55      0.38      0.44        16

   micro avg       0.82      0.82      0.82        82
   macro avg       0.70      0.65      0.67        82
weighted avg       0.80      0.82      0.80        82

0.8170731707317073
```

Demo Prediction

In [11]:
```
demo=[[10,29.5,1,4,4.1,6,48,5,6,19,17,70,48,1026.1,1022.7,7,8,14.1,18.9,0],[8,29.5,1,5,4.1,6,58,5,7,19,17,70,48,1026.1,1022.7,8,
y=regressor.predict(demo)
int(round(y[1],0))
```

Out[11]: 0

In [ ]:

RANDOM FOREST IMPLEMENTATION

CODE USED:

```python
# Random Forest Algorithm on Weather Dataset
from random import seed
from random import randrange
from csv import reader
from math import sqrt

# Load a CSV file
def load_csv(filename):
    dataset = list()
    with open(filename, 'r') as file:
        csv_reader = reader(file)
        for row in csv_reader:
            if not row:
                continue
            dataset.append(row)
    return dataset

# Convert string column to float
def str_column_to_float(dataset, column):
        for row in dataset:
                row[column] = float(row[column].strip())

# Convert string column to integer
def str_column_to_int(dataset, column):
        class_values = [row[column] for row in dataset]
        unique = set(class_values)
        lookup = dict()
        for i, value in enumerate(unique):
```

```
        lookup[value] = i
    for row in dataset:
        row[column] = lookup[row[column]]
    return lookup
```

 # Split a dataset into k folds

```
def cross_validation_split(dataset, n_folds):
    dataset_split = list()
    dataset_copy = list(dataset)
    fold_size = int(len(dataset) / n_folds)
    for i in range(n_folds):
        fold = list()
        while len(fold) < fold_size:
            index = randrange(len(dataset_copy))
            fold.append(dataset_copy.pop(index))
        dataset_split.append(fold)
    return dataset_split
```

 # Calculate accuracy percentage

```
def accuracy_metric(actual, predicted):
    correct = 0
    for i in range(len(actual)):
        if actual[i] == predicted[i]:
            correct += 1
    return correct / float(len(actual)) * 100.0
```

 # Evaluate an algorithm using a cross validation split

```
def evaluate_algorithm(dataset, algorithm, n_folds, *args):
```

```
	folds = cross_validation_split(dataset, n_folds)
	scores = list()
	for fold in folds:
		train_set = list(folds)
		train_set.remove(fold)
		train_set = sum(train_set, [])
		test_set = list()
		for row in fold:
			row_copy = list(row)
			test_set.append(row_copy)
			row_copy[-1] = None
		predicted = algorithm(train_set, test_set, *args)
		actual = [row[-1] for row in fold]
		accuracy = accuracy_metric(actual, predicted)
		scores.append(accuracy)
	return scores
```

 # Split a dataset based on an attribute and an attribute value

```
def test_split(index, value, dataset):
	left, right = list(), list()
	for row in dataset:
		if row[index] < value:
			left.append(row)
		else:
			right.append(row)
	return left, right
```

 # Calculate the Gini index for a split dataset

```
def gini_index(groups, classes):
    # count all samples at split point
    n_instances = float(sum([len(group) for group in groups]))
    # sum weighted Gini index for each group
    gini = 0.0
    for group in groups:
        size = float(len(group))
        # avoid divide by zero
        if size == 0:
            continue
        score = 0.0
        # score the group based on the score for each class
        for class_val in classes:
            p = [row[-1] for row in group].count(class_val) / size
            score += p * p
        # weight the group score by its relative size
        gini += (1.0 - score) * (size / n_instances)
    return gini
```

 # Select the best split point for a dataset

```
def get_split(dataset, n_features):
    class_values = list(set(row[-1] for row in dataset))
    b_index, b_value, b_score, b_groups = 999, 999, 999, None
    features = list()
    while len(features) < n_features:
        index = randrange(len(dataset[0])-1)
        if index not in features:
            features.append(index)
```

```
	for index in features:
		for row in dataset:
			groups = test_split(index, row[index], dataset)
			gini = gini_index(groups, class_values)
			if gini < b_score:
				b_index, b_value, b_score, b_groups = index, row[index], gini, groups
	return {'index':b_index, 'value':b_value, 'groups':b_groups}



# Create a terminal node value
def to_terminal(group):
	outcomes = [row[-1] for row in group]
	return max(set(outcomes), key=outcomes.count)

# Create child splits for a node or make terminal
def split(node, max_depth, min_size, n_features, depth):
	left, right = node['groups']
	del(node['groups'])
	# check for a no split
	if not left or not right:
		node['left'] = node['right'] = to_terminal(left + right)
		return
	# check for max depth
	if depth >= max_depth:
		node['left'], node['right'] = to_terminal(left), to_terminal(right)
		return
	# process left child
```

```
        if len(left) <= min_size:
            node['left'] = to_terminal(left)
        else:
            node['left'] = get_split(left, n_features)
            split(node['left'], max_depth, min_size, n_features, depth+1)
        # process right child
        if len(right) <= min_size:
            node['right'] = to_terminal(right)
        else:
      node['right'] = get_split(right, n_features)
      split(node['right'], max_depth, min_size, n_features, depth+1)
```

 # Build a decision tree

```
def build_tree(train, max_depth, min_size, n_features):
    root = get_split(train, n_features)
    split(root, max_depth, min_size, n_features, 1)
    return root
```

 # Make a prediction with a decision tree

```
def predict(node, row):
    if row[node['index']] < node['value']:
        if isinstance(node['left'], dict):
        return predict(node['left'], row)
        else:
            return node['left']
    else:
        if isinstance(node['right'], dict):
            return predict(node['right'], row)
```

```python
        else:
            return node['right']


# Create a random subsample from the dataset with replacement
def subsample(dataset, ratio):
    sample = list()
    n_sample = round(len(dataset) * ratio)
    while len(sample) < n_sample:
        index = randrange(len(dataset))
        sample.append(dataset[index])
    return sample

# Make a prediction with a list of bagged trees
def bagging_predict(trees, row):
    predictions = [predict(tree, row) for tree in trees]
    return max(set(predictions), key=predictions.count)

# Random Forest Algorithm
def random_forest(train, test, max_depth, min_size, sample_size, n_trees, n_features):
    trees = list()
    for i in range(n_trees):
        sample = subsample(train, sample_size)
        tree = build_tree(sample, max_depth, min_size, n_features)
        trees.append(tree)
    predictions = [bagging_predict(trees, row) for row in test]
    return(predictions)
```

 # Test the random forest algorithm

```
seed(2)

# load and prepare data

filename = 'weather_processed.csv'

dataset = load_csv(filename)

# convert string attributes to integers

for i in range(0, len(dataset[0])-1):

    str_column_to_float(dataset, i)

# convert class column to integers

str_column_to_int(dataset, len(dataset[0])-1)

# evaluate algorithm

n_folds = 5

max_depth = 10

min_size = 1

sample_size = 1.0

n_features = int(sqrt(len(dataset[0])-1))

for n_trees in [1, 5, 10]:

    scores = evaluate_algorithm(dataset, random_forest, n_folds, max_depth, min_size,
sample_size, n_trees, n_features)

    print('Trees: %d' % n_trees)

    print('Scores: %s' % scores)

    print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
```

OUTPUT:

```
Trees: 1
Scores: [83.07692307692308, 70.76923076923077, 75.38461538461539, 81.53846153846153, 78.46153846153847]
Mean Accuracy: 77.846%
Trees: 5
Scores: [90.76923076923077, 86.15384615384616, 86.15384615384616, 86.15384615384616, 78.46153846153847]
Mean Accuracy: 85.538%
Trees: 10
Scores: [90.76923076923077, 84.61538461538461, 81.53846153846153, 89.23076923076924, 86.15384615384616]
Mean Accuracy: 86.462%
```

CONCLUSION AND FUTURE WORK

Here by applying certain algorithms like decision tree, k-mean and random forest We can conclude that whether the rain going to be happen tomorrow or not. Here, Random forest is the most accurate algorithm which gives the accuracy of 81% whereas decision tree & k-mean algorithms are having lesser percentage of accuracy. we can predict whether it is going to rain on that day or not. We had done the prediction of rain on the basis of different factors that are responsible for affecting the rain conditions. Here the factors have their different impact on the prediction. Some of them are having higher impact & some have lesser impact. This shows that the factors which have lesser impact can be neglected. Here the factors like windspeed, winddirection, humidity are more impactable and also most important as concerning future work regarding it we can move on from machine learning to deep learning models, as well as we can apply a **STACKED** model consisting of the various models we have implemented above