1. INTRODUCTION

1.1 Overview

Rainfall has been a major concern these days. Weather conditions have been changing for time being. Rainfall forecasting is important otherwise, it may lead to many disasters. Irregular heavy rainfall may lead to the destruction of crops, heavy floods that can cause harm to human life. It is important to exactly determine the rainfall for effective use of water resources, crop productivity, and pre-planning of water structures.

This comparative study is conducted concentrating on the following aspects: modeling inputs, Visualising the data, modeling methods, and pre-processing techniques. The results provide a comparison of various evaluation metrics of these machine learning techniques and their reliability to predict rainfall by analyzing the weather data.

We will be using classification algorithms such as Decision tree, Random forest, KNN, and xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. Once the model is saved, we integrate it with flask application and also deploy the model in IBM.

1.2 Purpose

In today's situation, rainfall is considered to be one of the sole responsible factors for most of the significant things across the world. In India, agriculture is considered to be one of the important factors for deciding the economy of the country and agriculture is solely dependent on rainfall. In some of the areas which have water scarcity, to establish rain water harvester, prior prediction of the rainfall should be done. It will be very helpful if we could plan the availability of rain.

The objective of this project was to predict rain based upon historical weather data. This was approached as a binary classification problem, with the ultimate question being “will it rain tomorrow ?” answered by either yes or no. Multiple models were built to explore methods of predicting our response variable, RainTomorrow.

2. LITERATURE ANALYSIS

2.1 Existing problem

The existing Rainfall prediction model based on several ANN based architecture have been proposed to predict rainfall.

ANN based rainfall prediction has been reported where four years data has been used to predict rainfall one to three hours ahead. The prediction model was based on meteorological parameters such as wet bulb temperature, air pressure, relative humidity and cloudiness. The authors have found that wet bulb temperature could be the deciding factor in prediction of rainfall.

The learning algorithm tries to find out the optimal set of weights for the neural connections of the ANN. Thus, the training phase can be thought of as an optimizing problem where an error function is usually minimized.

It has been revealed that the standard algorithms may be unable to approximate the exact pattern of the data if is reasonably complex.

2.2 Proposed Solution

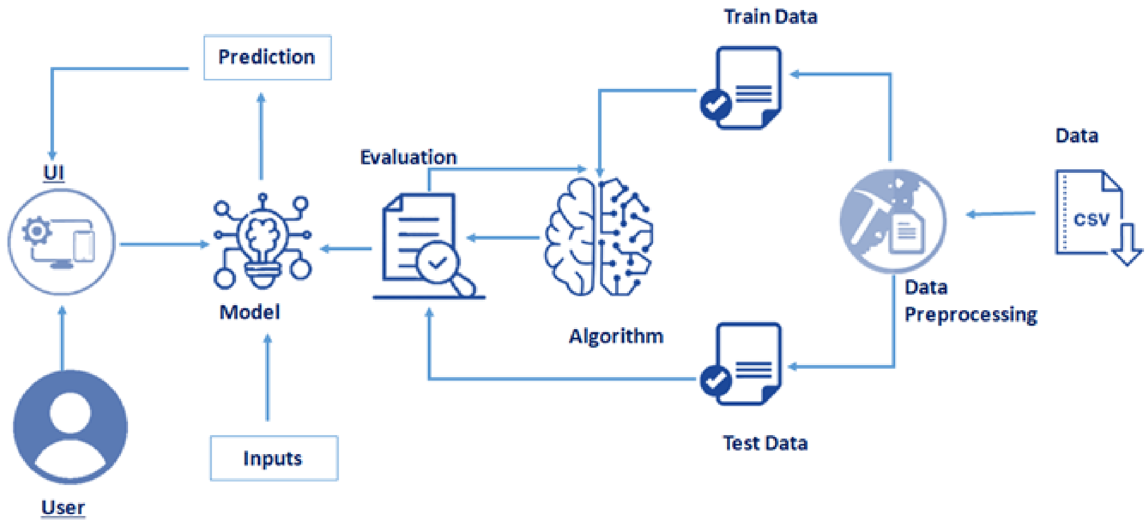
Currently, rainfall prediction has become one of the key factors for most of the water conservation systems in and across country. One of the biggest challenges is the complexity present in rainfall data. Most of the rainfall prediction system, nowadays are unable to find the hidden layers or any non-linear patterns present in the system. This project will assist to find all the hidden layers as well as non-linear patterns, which is useful for performing the precise prediction of rainfall

Rainfall prediction is the application to predict the rainfall in a given region. It can be done in two types. The first is to analyze the physical law that affects rainfall and the second one is to make a system which will discover hidden patterns or the features that affects the physical factors and the process involved in achieving it. The second one is better because it doesn’t include any type of mathematical calculations and can be useful for complex and non-linear data

Proposed system predict rainfall based on rainfall based on a larger data set. Here instead of focusing on a model in uses a number of models that has been tested to find the accuracy and selects the best one which gives the accurate result.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software designing

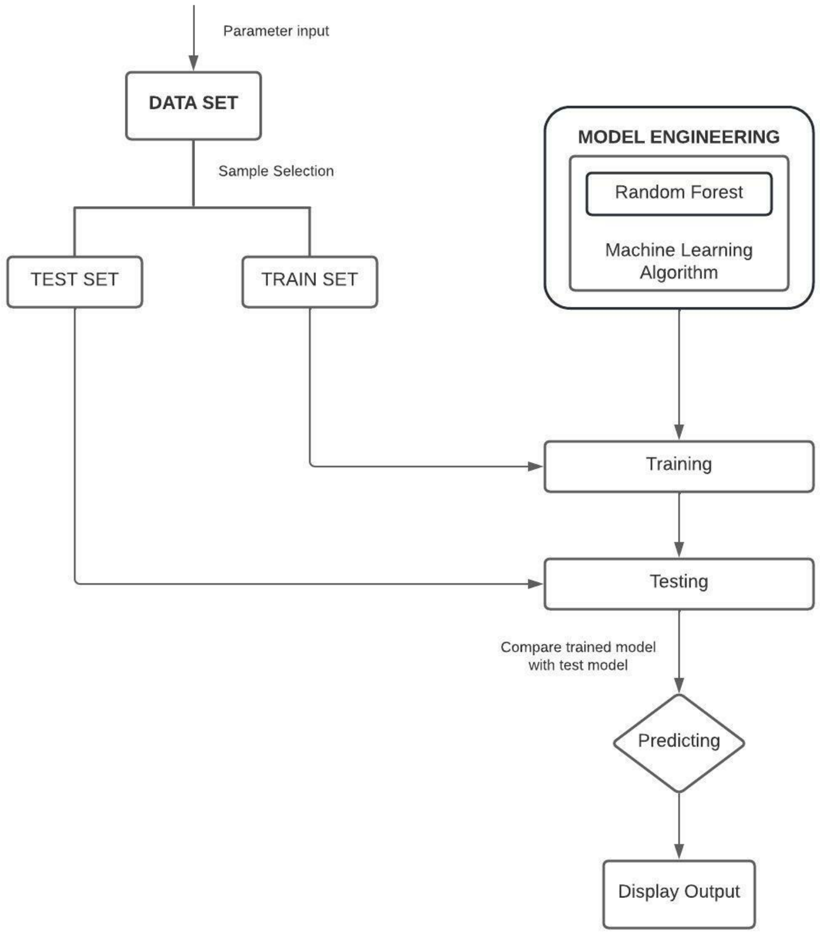
Processor	:	intel i5 7th Gen or above
Ram	:	4GB
Hard disk	:	100GB
Input device	:	Standard Keyboard and Mouse
Output device	:	Monitor
Operating System	:	Windows
Programming	:	python 3.6

4. EXPERIMENTAL INVESTIGATION

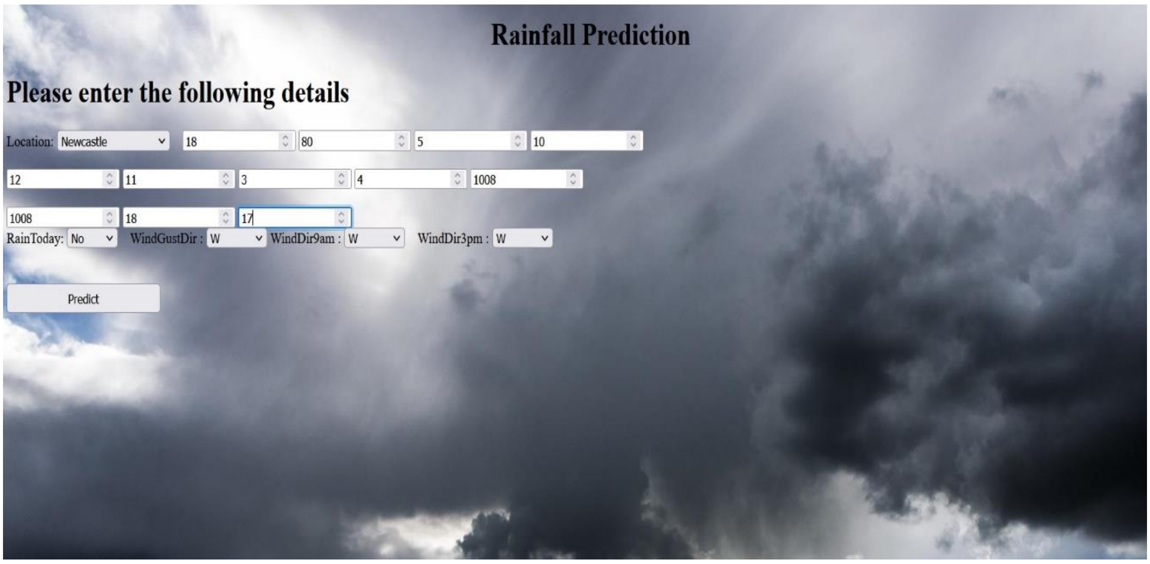
As the data we have collected is too large the major task in this project was to find the correctness of it. After collecting the data building the model and find the best model with more accuracy is the next step, after the model has been setup the next thing is to identify whether it produces the correct result as we are calculating on whether it will rain tomorrow or not we have to check it for a couple of weeks because we cannot decide it by checking it with one or two days.

Even though we have got the correct result for one or two days we cannot say that it produces the correct result all the time for that we have to run the test for a few days to weeks. That was the next challenge in our project.

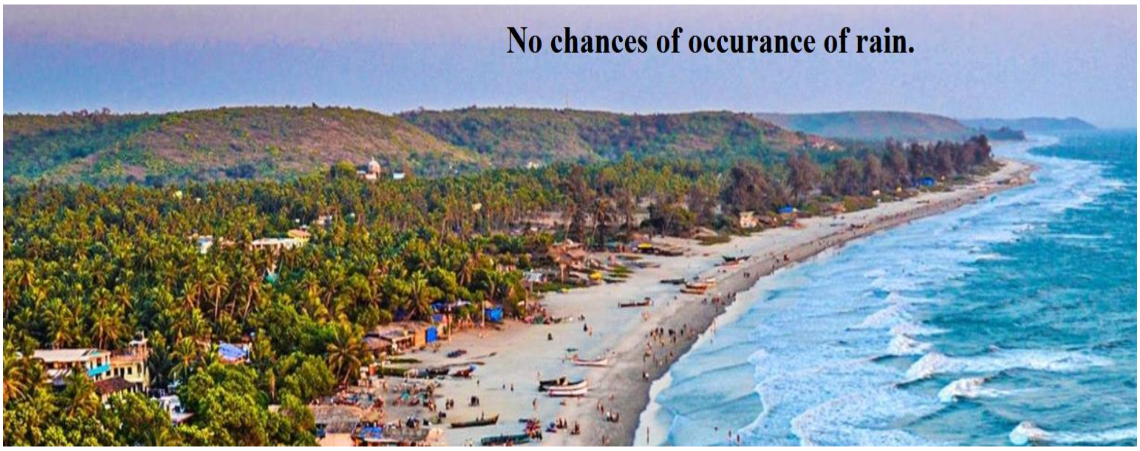
5. FLOWCHART


Parameter input
DATA SET
Sample Selection
TEST SET
TRAIN SET
MODEL ENGINEERING
Random Forest
Machine Learning Algorithm
Training
Testing
Compare trained model with test model
Predicting
Display Output

6. RESULT


Rainfall Prediction
Please enter the following details
Location: Newcastle
18
80
5
10
12
11
3
4
1008
1008
18
17
RainToday: No
WindGustDir : W
WindDir9am : W
WindDir3pm : W
Predict


chances of rain today.


No chances of occurance of rain.

7. ADVANTAGES AND DISADVANTAGES

- It produces more accurate result that the existing system.
- As we can know about the rainfall ahead of time we can plan things earlier
- It provides a more user friendly interface than the existing system.
- This system was designed to predict the condition of tomorrow, i.e whether it will rain tomorrow or not. So it is concerned with a day to day experiment
- It requires a little bit more information as the input, based on which the prediction is made
- As it is a prediction project we cannot say that it is 100% accurate. Errors can occur.

8. APPLICATIONS

Rainfall prediction system is every much helpful in weather forecasting field. One of the major problem faced today is the inaccuracy in weather forecasting, so this will be very useful in the forecasting fields. Agriculture is the main field that is being affected by the rainfall, continuous and unpredictable rainfall can make a huge loss in the agriculture sector similarly decreasing amount of rainfall can also have a huge impact on the agriculture field, so they have to be properly informed about these to take necessary actions to withstand the effect of rainfall.

9. CONCLUSION

Rainfall being one of the sole responsibilities for maximum economy of India, it should be considered the primary concern for most of us. The current approach for rainfall prediction fails in most of the complex cases, as it is unable to predict the hidden layers present, which is yet to be recognized for performing the precise prediction. Through this project we will able to predict the more accurate result for the rainfall, as it has been built by considering many models and test all of them using a large amount of data, after analysing the result the model which gives the accurate result has been used to develop this project. Through this we can get a more accurate prediction.

10. FUTURE SCOPE

The future enhancement of this project can be an approach towards about how to reduce the percentage of errors present. Along with that one of the major enhancements will be to decrease the ratio for train data to test data, so that it will assist in improving the level of prediction within the available time and complexity. The accuracy of the algorithm can be additionally tested on increase in the complexity. Many other types of errors can be calculated in order to test the accuracy of any of the above algorithms. Henceforth, algorithm for testing daily basis dataset instead of accumulated dataset could be of paramount Importance for further research.

11. BIBLIOGRAPHY

1. Janani, B; Sebastian, P. (2014). Analysis on the weather forecasting and techniques. International Journal of Advanced Research in Computer Engineering & Technology, 3(1), 59–61.

2. Chaudhari, M. S., & Choudhari, N. K. (2017). Open Access Study of Various Rainfall Estimation & Prediction Techniques Using Data Mining. American Journal of Engineering Research (AJER), 7, 137–139.

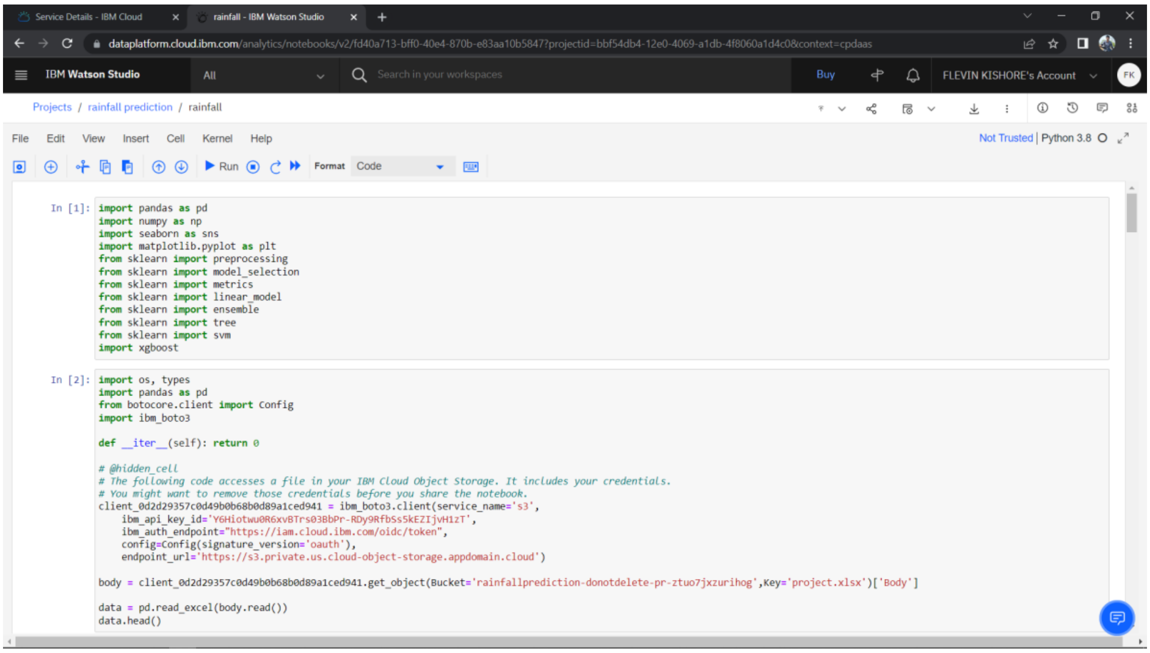
3. Aakash Parmar, Kinjal Mistree, M. S. (2017). Machine Learning Techniques for rainfall prediction: A Review. International Conference on Innovations in Information Embedded and Communication Systems (ICIIECS).

4. Aftab, S., Ahmad, M., Hameed, N., Bashir, M. S., Ali, I., & Nawaz, Z. (2018). Rainfall prediction using data mining techniques: A systematic literature review. International Journal of Advanced Computer Science and Applications, 9(5), 143–150.

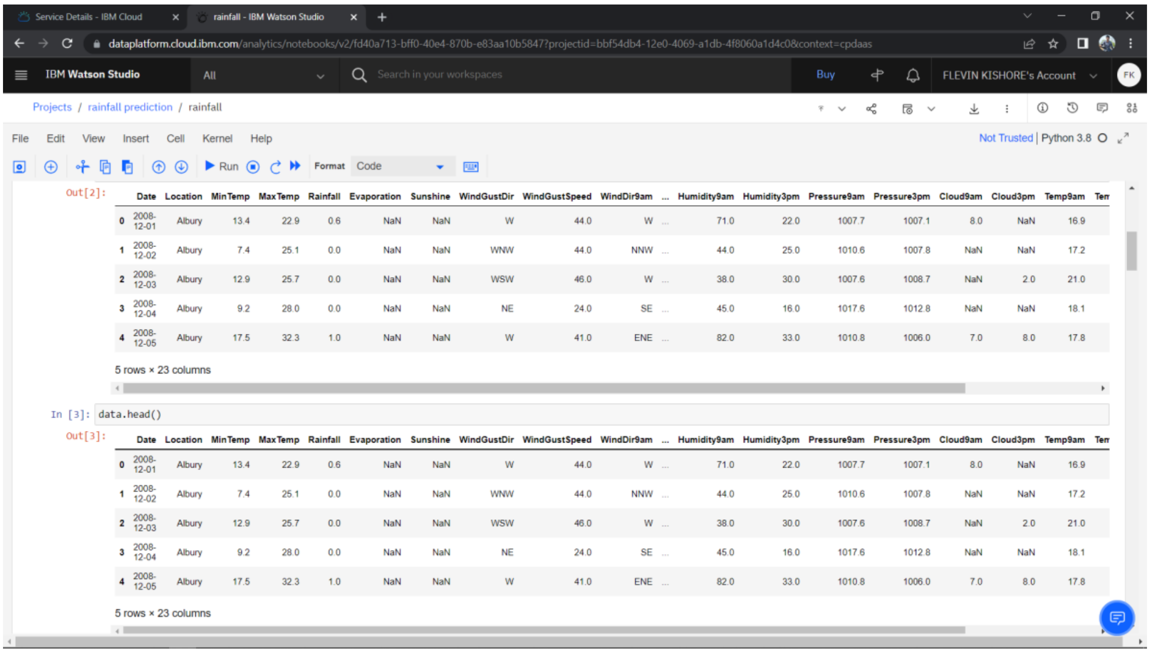
5. Meng – Hua Yen , Ding – Wei Liu , Yi – Chia Hsin, C. – E. L. and C. – C. C. (2019). Application of the deep learning for the prediction of rainfall in Southern Taiwan. Scientific Reports, 9(1), 1–9.

APPENDIX

A. Source Code



```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn import model_selection
from sklearn import metrics
from sklearn import linear_model
from sklearn import ensemble
from sklearn import tree
from sklearn import svm
import xgboost

In [2]: import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_0d2d29357c0d49b0b68b0d89a1ced941 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Y6HiotwuOR6xvBTrs03BbPr-RDy9RfbSs5kEZIjvH1zT',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

body = client_0d2d29357c0d49b0b68b0d89a1ced941.get_object(Bucket='rainfallprediction-donotdelete-pr-ztuo7jxzurihog',Key='project.xlsx')['Body']

data = pd.read_excel(body.read())
data.head()
```



Out[2]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Tem
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	1007.7	1007.1	8.0	NaN	16.9	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	1010.6	1007.8	NaN	NaN	17.2	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	1007.6	1008.7	NaN	2.0	21.0	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	1017.6	1012.8	NaN	NaN	18.1	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	1010.8	1006.0	7.0	8.0	17.8	

5 rows × 23 columns

```
In [3]: data.head()
```

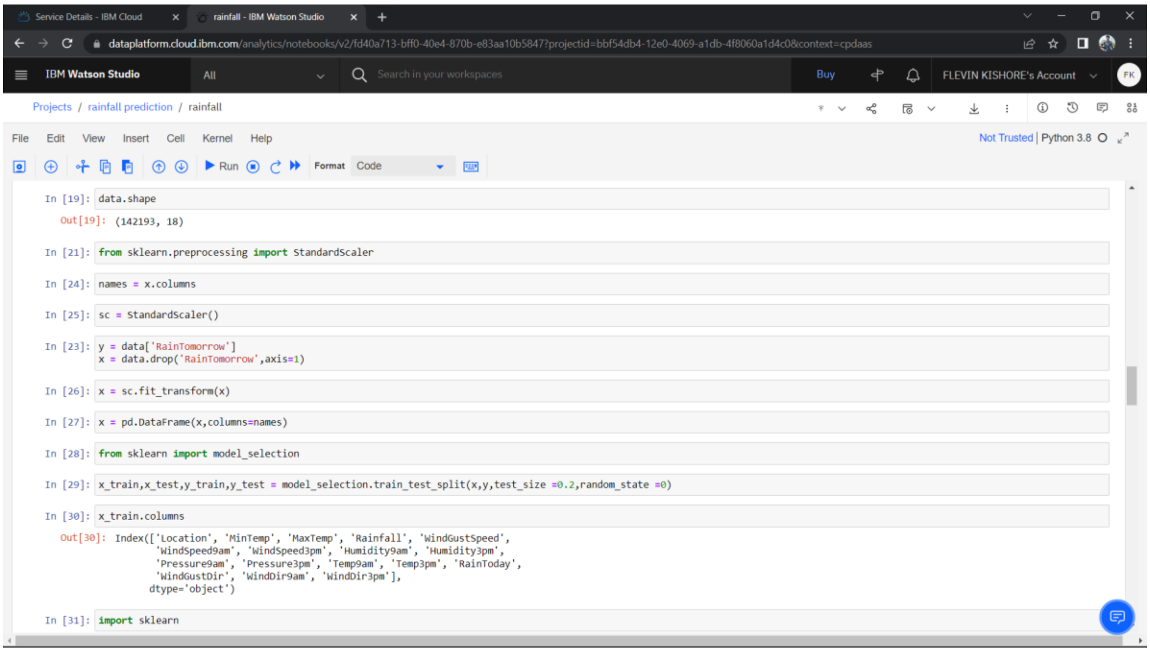
Out[3]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Tem
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	1007.7	1007.1	8.0	NaN	16.9	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	1010.6	1007.8	NaN	NaN	17.2	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	1007.6	1008.7	NaN	2.0	21.0	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	1017.6	1012.8	NaN	NaN	18.1	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	1010.8	1006.0	7.0	8.0	17.8	

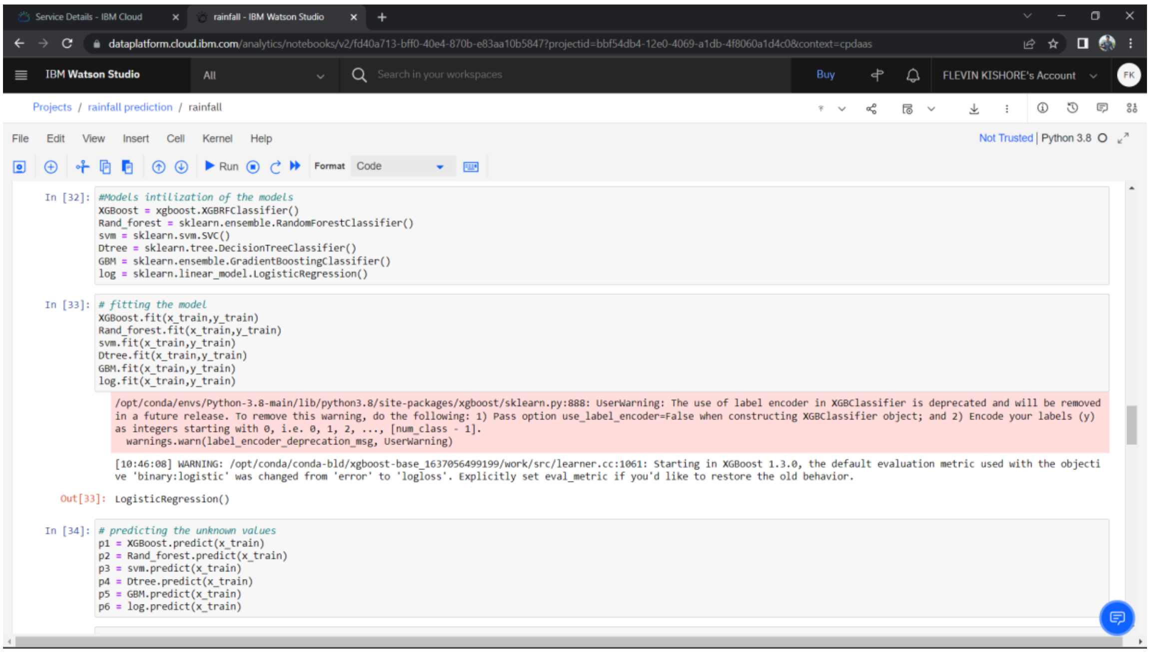
5 rows × 23 columns

```
In [5]: # removing columns with more than 20% missing values and segregatting cat and num variables
        data_cat = data[['RainToday','WindGustDir','WindDir9am','WindDir3pm']]
        data.drop(columns=['Evaporation','Sunshine','Cloud9am','Cloud3pm'],axis=1,inplace=True)
        data.drop(columns=['RainToday','WindGustDir','WindDir9am','WindDir3pm'],axis=1,inplace=True)

In [6]: # filling the missing data of numeric variables with mean
        data['MinTemp'].fillna(data['MinTemp'].mean(),inplace=True)
        data['MaxTemp'].fillna(data['MaxTemp'].mean(),inplace=True)
        data['Rainfall'].fillna(data['Rainfall'].mean(),inplace=True)
        data['WindGustSpeed'].fillna(data['WindGustSpeed'].mean(),inplace=True)
        data['WindSpeed9am'].fillna(data['WindSpeed9am'].mean(),inplace=True)
        data['WindSpeed3pm'].fillna(data['WindSpeed3pm'].mean(),inplace=True)
        data['Humidity9am'].fillna(data['Humidity9am'].mean(),inplace=True)
        data['Humidity3pm'].fillna(data['Humidity3pm'].mean(),inplace=True)
        data['Pressure9am'].fillna(data['Pressure9am'].mean(),inplace=True)
        data['Pressure3pm'].fillna(data['Pressure3pm'].mean(),inplace=True)
        data['Temp9am'].fillna(data['Temp9am'].mean(),inplace=True)
        data['Temp3pm'].fillna(data['Temp3pm'].mean(),inplace=True)

In [7]: # loading the names of categorical columns
        cat_names = data_cat.columns

In [8]: # intializing the simple imputer for missing categorical values
        import numpy as np
        from sklearn.impute import SimpleImputer
        imp_mode = SimpleImputer(missing_values=np.nan, strategy='most_frequent')

In [9]: # fitting and transforming the missing data
        data_cat = imp_mode.fit_transform(data_cat)

In [10]: # converting array to dataframe
         data_cat = pd.DataFrame(data_cat,columns=cat_names)
```

```
In [11]: # concatinating the categorical and numeric data
         data = pd.concat([data,data_cat],axis=1)

In [12]: data.shape

Out[12]: (145460, 19)

In [13]: #importing the labelencoder
         from sklearn.preprocessing import LabelEncoder

In [14]: le = LabelEncoder()

In [15]: #fitting and transforming the categorical data
         data['Location'] = le.fit_transform(data['Location'])
         data['RainToday'] = le.fit_transform(data['RainToday'])
         data['WindGustDir'] = le.fit_transform(data['WindGustDir'])
         data['WindDir9am'] = le.fit_transform(data['WindDir9am'])
         data['WindDir3pm'] = le.fit_transform(data['WindDir3pm'])

In [16]: data['Date'].unique()

Out[16]: array(['2008-12-01T00:00:00.000000000', '2008-12-02T00:00:00.000000000',
                '2008-12-03T00:00:00.000000000', ...,
                '2008-01-29T00:00:00.000000000', '2008-01-30T00:00:00.000000000',
                '2008-01-31T00:00:00.000000000'], dtype='datetime64[ns]')

In [17]: # removing the main column
         data.drop(['Date'],axis=1,inplace=True)

In [18]: data.dropna(axis=0,how='any',inplace=True)
```



```
In [19]: data.shape
Out[19]: (142193, 18)

In [21]: from sklearn.preprocessing import StandardScaler

In [24]: names = x.columns

In [25]: sc = StandardScaler()

In [23]: y = data['RainTomorrow']
         x = data.drop('RainTomorrow',axis=1)

In [26]: x = sc.fit_transform(x)

In [27]: x = pd.DataFrame(x,columns=names)

In [28]: from sklearn import model_selection

In [29]: x_train,x_test,y_train,y_test = model_selection.train_test_split(x,y,test_size =0.2,random_state =0)

In [30]: x_train.columns
Out[30]: Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed',
                'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
                'Pressure9am', 'Pressure3pm', 'Temp9am', 'Temp3pm', 'RainToday',
                'WindGustDir', 'WindDir9am', 'WindDir3pm'],
               dtype='object')

In [31]: import sklearn
```



```
In [32]: #Models intilization of the models
         XGBoost = xgboost.XGBRFClassifier()
         Rand_forest = sklearn.ensemble.RandomForestClassifier()
         svm = sklearn.svm.SVC()
         Dtree = sklearn.tree.DecisionTreeClassifier()
         GBM = sklearn.ensemble.GradientBoostingClassifier()
         log = sklearn.linear_model.LogisticRegression()

In [33]: # fitting the model
         XGBoost.fit(x_train,y_train)
         Rand_forest.fit(x_train,y_train)
         svm.fit(x_train,y_train)
         Dtree.fit(x_train,y_train)
         GBM.fit(x_train,y_train)
         log.fit(x_train,y_train)

/opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages/xgboost/sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:46:08] WARNING: /opt/conda/conda-bld/xgboost-base_1637056499199/work/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

Out[33]: LogisticRegression()

In [34]: # predicting the unknown values
         p1 = XGBoost.predict(x_train)
         p2 = Rand_forest.predict(x_train)
         p3 = svm.predict(x_train)
         p4 = Dtree.predict(x_train)
         p5 = GBM.predict(x_train)
         p6 = log.predict(x_train)
```

Service Details - IBM Cloud | rainfall - IBM Watson Studio

dataplatform.cloud.ibm.com/analytics/notebooks/v2/fd40a713-bff0-40e4-870b-e83aa10b5847/view?projectid=bbf54db4-12e0-4069-a1db-4f8060a1d4c0&context=cpdaas

IBM Watson Studio | All | Search in your workspaces | Buy | FLEVIN KISHORE's Account

Projects / rainfall prediction / rainfall

```
In [35]: #checking the accuraccy score
         print("xgboost:",metrics.accuracy_score(y_train,p1))
         print("Rand_forest:",metrics.accuracy_score(y_train,p2))
         print("svm:",metrics.accuracy_score(y_train,p3))
         print("Dtree:",metrics.accuracy_score(y_train,p4))
         print("GBM:",metrics.accuracy_score(y_train,p5))
         print("log:",metrics.accuracy_score(y_train,p6))

         xgboost: 0.8425989415756808
         Rand_forest: 0.9999472546020359
         svm: 0.8550116918965487
         Dtree: 0.9999648364013574
         GBM: 0.849148161822881
         log: 0.8384056824375407

In [36]: # predicting the test unknown values
         t1 = XGBoost.predict(x_test)
         t2 = Rand_forest.predict(x_test)
         t3 = svm.predict(x_test)
         t4 = Dtree.predict(x_test)
         t5 = GBM.predict(x_test)
         t6 = log.predict(x_test)

In [37]: print("xgboost:",metrics.accuracy_score(y_test,t1))
         print("Rand_forest:",metrics.accuracy_score(y_test,t2))
         print("svm:",metrics.accuracy_score(y_test,t3))
         print("Dtree:",metrics.accuracy_score(y_test,t4))
         print("GBM:",metrics.accuracy_score(y_test,t5))
         print("log:",metrics.accuracy_score(y_test,t6))

         xgboost: 0.8426104996659517
         Rand_forest: 0.8568515067337108
         svm: 0.8518935264953057
         Dtree: 0.7769612152326031
         GBM: 0.8505924962199796
         log: 0.8420478919793242

In [38]: model = ensemble.RandomForestClassifier(criterion='entropy',max_features='sqrt',n_estimators = 25)
```

Service Details - IBM Cloud | rainfall - IBM Watson Studio

dataplatform.cloud.ibm.com/analytics/notebooks/v2/fd40a713-bff0-40e4-870b-e83aa10b5847/view?projectid=bbf54db4-12e0-4069-a1db-4f8060a1d4c0&context=cpdaas

IBM Watson Studio | All | Search in your workspaces | Buy | FLEVIN KISHORE's Account

Projects / rainfall prediction / rainfall

```
In [39]: model.fit(x_train,y_train)
         y_tp= model.predict(x_train)
         sklearn.metrics.accuracy_score(y_train,y_tp)

Out[39]: 0.9983824744624364

In [40]: metrics.confusion_matrix(y_train,y_tp)

Out[40]: array([[88230,    19],
                [  165, 25340]])

In [41]: y_pred= model.predict(x_test)
         sklearn.metrics.accuracy_score(y_test,y_pred)

Out[41]: 0.8532297197510461

In [42]: import ibm_watson_machine_learning

In [43]: from ibm_watson_machine_learning import APIClient
         import json
         import numpy as np

In [56]: wml_credentials = {
             "apikey" : "9417L-N8ej71KwqkN_GQdwUO5hFeBAOJO0FasObjrJ_t",
             "url" : "https://us-south.ml.cloud.ibm.com"
         }

In [45]: wml_client = APIClient(wml_credentials)
         wml_client.spaces.list()

         Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
         ------------------------------------  ------------------  ------------------------
         ID                                    NAME                CREATED
         c403925f-55b5-42a7-a5f8-b966abc486c8  rainfall_prediction 2022-03-05T09:44:21.021Z
         ------------------------------------  ------------------  ------------------------

In [46]: SPACE_ID = "c403925f-55b5-42a7-a5f8-b966abc486c8"
```

Service Details - IBM Cloud | rainfall - IBM Watson Studio

dataplatform.cloud.ibm.com/analytics/notebooks/v2/fd40a713-bff0-40e4-870b-e83aa10b5847/view?projectid=bbf54db4-12e0-4069-a1db-4f8060a1d4c0&context=cpdaas

IBM Watson Studio | All | Search in your workspaces | Buy | FLEVIN KISHORE's Account

Projects / rainfall prediction / rainfall

```
In [64]: deployment_props = {
             wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
             wml_client.deployments.ConfigurationMetaNames.ONLINE:{}
         }

In [65]: deployment = wml_client.deployments.create(artifact_uid=model_uid,meta_props=deployment_props)

         #######################################################################################

         Synchronous deployment creation for uid: '210db845-42ca-4638-8cda-a18a4afdc0c8' started

         #######################################################################################


         initializing
         Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

         ready


         ------------------------------------------------------------------------------------------------
         Successfully finished deployment creation, deployment_uid='31b8a4af-295b-4f63-8af1-935bf55cd75b'
         ------------------------------------------------------------------------------------------------


In [68]: deployment_uid = wml_client.deployments.get_uid(deployment)
         deployment_uid

Out[68]: '31b8a4af-295b-4f63-8af1-935bf55cd75b'

In [73]: payload= {"input data":[{"field":x_test.columns.to_numpy().tolist(),"values":x_test.to_numpy().tolist()}]}

In [ ]:
```