Rainfall Prediction - Weather Forecasting

Abstract:- Rainfall prediction is one of the challenging and uncertain tasks which has a significant impact on human society. Timely and accurate predictions can help to proactively reduce human and financial loss. This study presents a set of experiments which involve the use of prevalent machine learning techniques to build models to predict whether it is going to rain tomorrow or not based on weather data for that particular day in major cities of Australia. This comparative study is conducted concentrating on three aspects: modeling inputs, 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 the rainfall by analyzing the weather data.

Keywords: rainfall prediction, oversampling, undersampling, classifiers, model evaluation.

Submitted by:-

Abhishek V.S

Introduction:-

Rainfall prediction remains a serious concern and has attracted the attention of governments, industries, risk management entities, as well as the scientific community. Rainfall is a climatic factor that affects many human activities like agricultural production, construction, power generation, forestry and tourism, among others. To this extent, rainfall prediction is essential since this variable is the one with the highest correlation with adverse natural events such as landslides, flooding, mass movements and avalanches. These incidents have affected society for years. Therefore, having an appropriate approach for rainfall prediction makes it possible to take preventive and mitigation measures for these natural phenomena.

To solve this uncertainty, we used various machine learning techniques and models to make accurate and timely predictions. These paper aims to provide end to end machine learning life cycle right from Data preprocessing to implementing models to evaluating them. Data preprocessing steps include imputing missing values, feature transformation, encoding categorical features, feature scaling and feature selection. We implemented models such as Logistic Regression, Decision Tree, K Nearest Neighbour , and Ensembles.

For evaluation purpose we used Accuracy, Precision, Recall, F-Score and Area Under Curve as evaluation metrics. For our experiments, we train our classifiers using Australian weather data gathered from various weather stations in Australia from last 10 years.

Predict two things:-

1. Design a predictive model with the use of machine learning algorithms to forecast whether or not it will rain tomorrow.
2. Design a predictive model with the use of machine learning algorithms to predict how much rainfall could be there.

CASE STUDY:-

Date - The date of observation

Location -The common name of the location of the weather station

MinTemp -The minimum temperature in degrees celsius

MaxTemp -The maximum temperature in degrees celsius

Rainfall -The amount of rainfall recorded for the day in mm

Evaporation -The so-called Class A pan evaporation (mm) in the 24 hours to 9am

Sunshine -The number of hours of bright sunshine in the day.

WindGustDir- The direction of the strongest wind gust in the 24 hours to midnight

WindGustSpeed -The speed (km/h) of the strongest wind gust in the 24 hours to midnight

WindDir9am -Direction of the wind at 9am

WindDir3pm -Direction of the wind at 3pm

WindSpeed9am -Wind speed (km/hr) averaged over 10 minutes prior to 9am

WindSpeed3pm -Wind speed (km/hr) averaged over 10 minutes prior to 3pm

Humidity9am -Humidity (percent) at 9am

Humidity3pm -Humidity (percent) at 3pm

Pressure9am -Atmospheric pressure (hpa) reduced to mean sea level at 9am

Pressure3pm -Atmospheric pressure (hpa) reduced to mean sea level at 3pm

Cloud9am - Fraction of sky obscured by cloud at 9am.

Cloud3pm -Fraction of sky obscured by cloud

Temp9am-Temperature (degrees C) at 9am

Temp3pm -Temperature (degrees C) at 3pm

RainToday -Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0

RainTomorrow -The amount of next day rain in mm

Methodology

In this paper, the overall architecture include four major components: Data Exploration and Analysis, Data Pre-processing, Model Implementation, and Model Evaluation, as shown in Fig. 1.

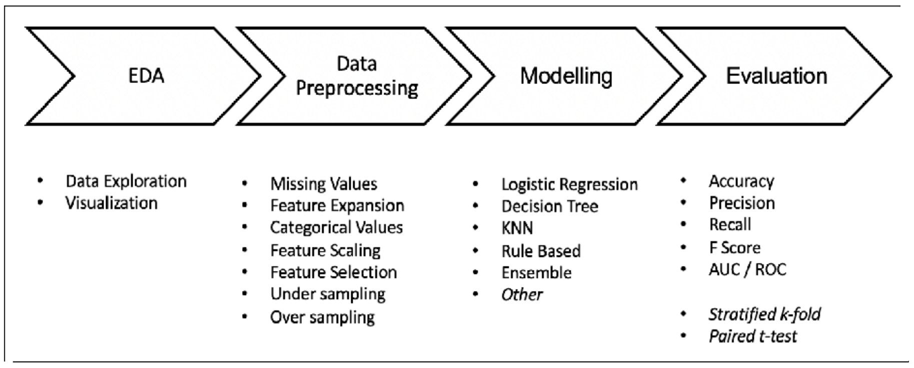
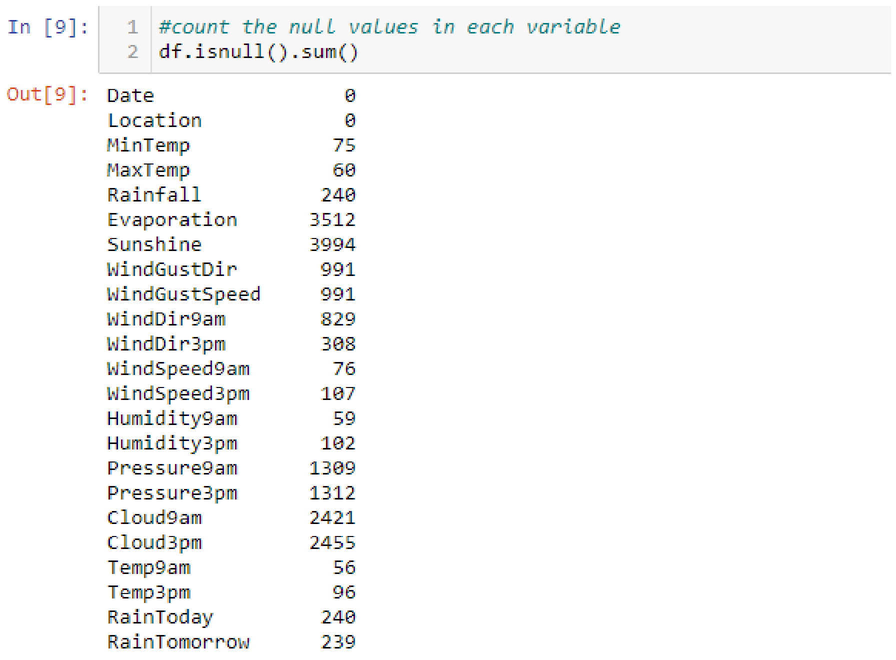


Fig. 1: Overall Architecture

1. Data Exploration and Analysis

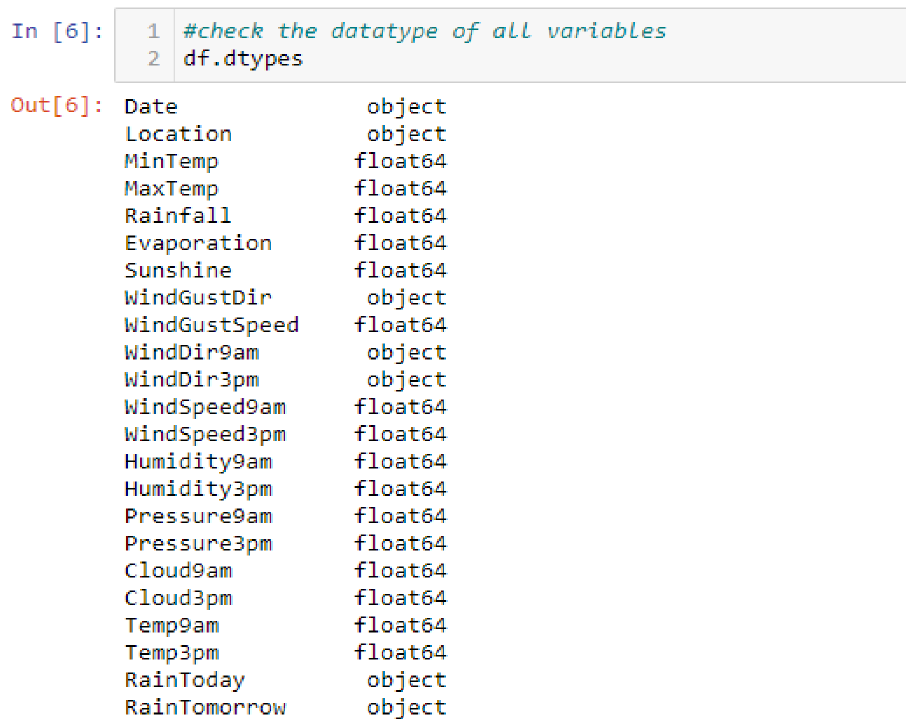
Exploratory Data Analysis is valuable to machine learning problems since it allows to get closer to the certainty that the future results will be valid, correctly interpreted, and applicable to the desired business contexts. Such level of certainty can be achieved only after raw data is validated and checked for anomalies, ensuring that the data set was collected without errors. EDA also helps to find insights that were not evident or worth investigating to business stakeholders and researchers.



```
In [9]:  #count the null values in each variable
         df.isnull().sum()

Out[9]:  Date                0
         Location            0
         MinTemp            75
         MaxTemp            60
         Rainfall          240
         Evaporation      3512
         Sunshine         3994
         WindGustDir       991
         WindGustSpeed     991
         WindDir9am        829
         WindDir3pm        308
         WindSpeed9am       76
         WindSpeed3pm      107
         Humidity9am        59
         Humidity3pm       102
         Pressure9am      1309
         Pressure3pm      1312
         Cloud9am         2421
         Cloud3pm         2455
         Temp9am            56
         Temp3pm            96
         RainToday         240
         RainTomorrow      239
```

Fig 2:- Null Values in Dataset



```
In [6]:  #check the datatype of all variables
         df.dtypes

Out[6]:  Date              object
         Location          object
         MinTemp          float64
         MaxTemp          float64
         Rainfall         float64
         Evaporation      float64
         Sunshine         float64
         WindGustDir       object
         WindGustSpeed    float64
         WindDir9am        object
         WindDir3pm        object
         WindSpeed9am     float64
         WindSpeed3pm     float64
         Humidity9am      float64
         Humidity3pm      float64
         Pressure9am      float64
         Pressure3pm      float64
         Cloud9am         float64
         Cloud3pm         float64
         Temp9am          float64
         Temp3pm          float64
         RainToday         object
         RainTomorrow      object
```

Fig 3:-Data type of All Variables in Dataset

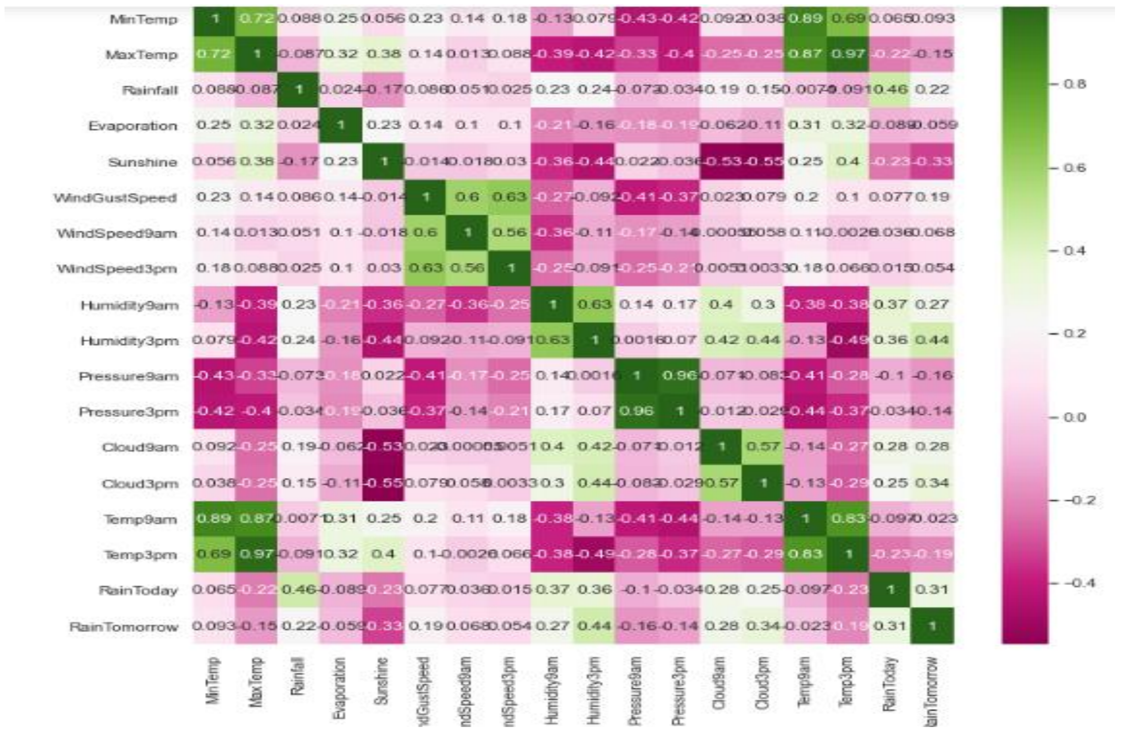


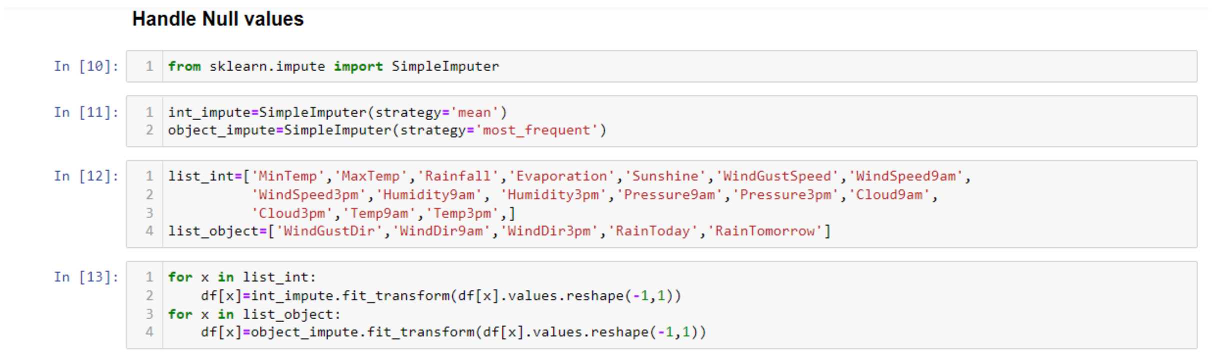
Fig 4:-Heat Map for Correlation

The correlation matrix depicts that the features - MaxTemp, Pressure9am, Pressure3pm, Temp3pm and Temp9am are negatively correlated with target variable.Hence, we can drop this features in our feature selection step later.

2. Data Pre-processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. We have carried below preprocessing steps.

Missing Values: As per our EDA step, we learned that we have few instances with null values. Hence, this becomes one of the important step. To impute the missing values, we will group our instances based on the location and date and thereby replace the null values by there respective mean values.



Handle Null values

```
In [10]:  from sklearn.impute import SimpleImputer

In [11]:  int_impute=SimpleImputer(strategy='mean')
          object_impute=SimpleImputer(strategy='most_frequent')

In [12]:  list_int=['MinTemp','MaxTemp','Rainfall','Evaporation','Sunshine','WindGustSpeed','WindSpeed9am',
                   'WindSpeed3pm','Humidity9am', 'Humidity3pm','Pressure9am','Pressure3pm','Cloud9am',
                   'Cloud3pm','Temp9am','Temp3pm',]
          list_object=['WindGustDir','WindDir9am','WindDir3pm','RainToday','RainTomorrow']

In [13]:  for x in list_int:
              df[x]=int_impute.fit_transform(df[x].values.reshape(-1,1))
          for x in list_object:
              df[x]=object_impute.fit_transform(df[x].values.reshape(-1,1))
```

Fig 5:-Handle Null values

Feature Selection: Feature Selection is the process where you automatically or manually select those features which contribute most to our prediction variable or output. Having irrelevant features in data can decrease the accuracy of the models and make the model learn based on irrelevant features. Feature selection helps to reduce over fitting, improves accuracy and reduces training time.

We used two techniques to perform this activity and got the same results.

- Univariate Selection: Statistical tests can be used to select those features that have the strongest relationship with the output variable. The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features. We used chi-squared statistical test for non-negative features to select 5 of the best features from our data set.
- **Correlation Matrix with Heatmap**: Correlation states how the features are related to each other or the target variable. Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable). Heatmap makes it easy to identify which features are most related to the target variable, we plotted heatmap of correlated features using the seaborn library(figure 4).

Handling Class Imbalance We learned in our EDA step that our data set is highly imbalanced. Imbalanced data results in biased results as our model doesn’t learn much about the minority class. We performed two experiments one with oversampled data and another with under sampled data.

- **Undersampling**: We used Imblearn’s random under sampler library to eliminate instances of majority class. This elimination is based on distance so that there is minimum loss of information.
- **Oversampling**: We used Imblearn’s SMOTE technique to generate synthetic instances for minority class. A subset of data is taken from the minority class as an example and then new synthetic similar instances are created.

Apply SMOTE

```
In [61]:  #Apply SMOTE bcz dataset is imbalanced to make Balanced apply SMOTE
          from imblearn.over_sampling import SMOTE

In [62]:  smt=SMOTE()

In [63]:  dfx=df1.drop('RainTomorrow',axis=1)
          dfy=df1['RainTomorrow']

In [64]:  train_x,train_y=smt.fit_resample(dfx,dfy)

In [65]:  #count plot of target variable
          sns.countplot(train_y)

Out[65]:  <AxesSubplot:xlabel='RainTomorrow', ylabel='count'>
```

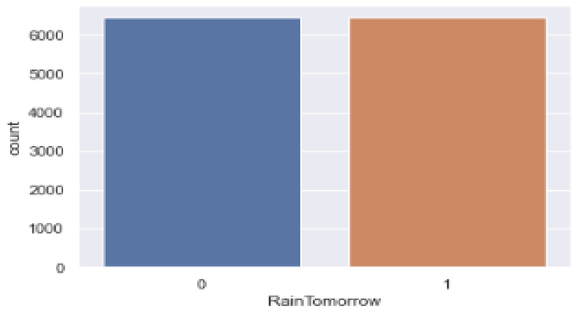
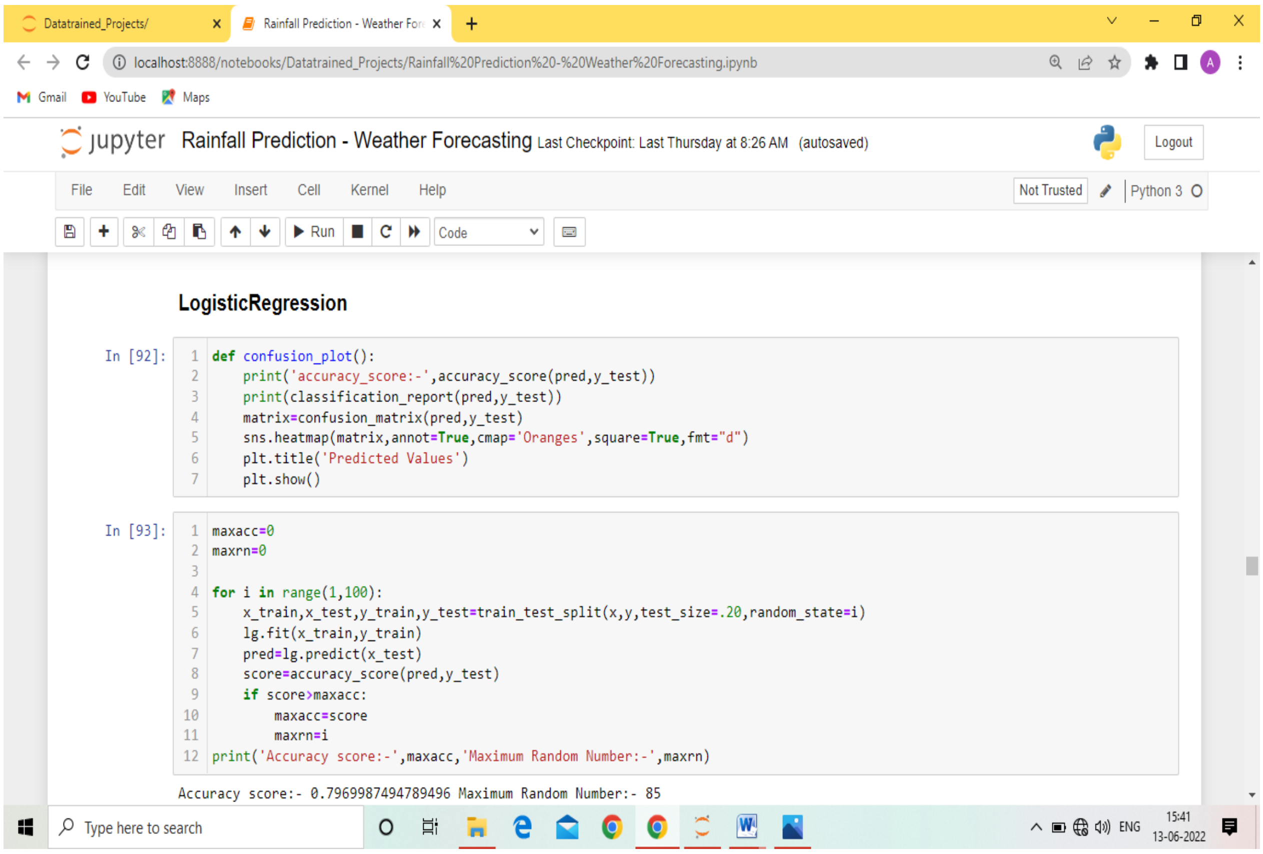


Fig 6:-SMOTE

3. Models

We chose different classifiers each belonging to different model family (such as Linear classifier, Tree-based, Distance-based and Ensemble). All the classifiers were implemented using **scikit-learn** except for Decision table which was implemented using **Jupyter**.
The following classification algorithms have been used to build prediction models to perform the experiments:

Logistic Regression is a classification algorithm used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. We can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logistic function. Hence, this makes Logistic Regression a better fit as ours isa binary classification problem.


Datatrained_Projects/
Rainfall Prediction - Weather For
localhost:8888/notebooks/Datatrained_Projects/Rainfall%20Prediction%20-%20Weather%20Forecasting.ipynb
Gmail
YouTube
Maps
jupyter
Rainfall Prediction - Weather Forecasting Last Checkpoint: Last Thursday at 8:26 AM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3
Run
Code
Type here to search
ENG
15:41
13-06-2022

LogisticRegression

In [92]:
```
def confusion_plot():
    print('accuracy_score:-',accuracy_score(pred,y_test))
    print(classification_report(pred,y_test))
    matrix=confusion_matrix(pred,y_test)
    sns.heatmap(matrix,annot=True,cmap='Oranges',square=True,fmt="d")
    plt.title('Predicted Values')
    plt.show()
```

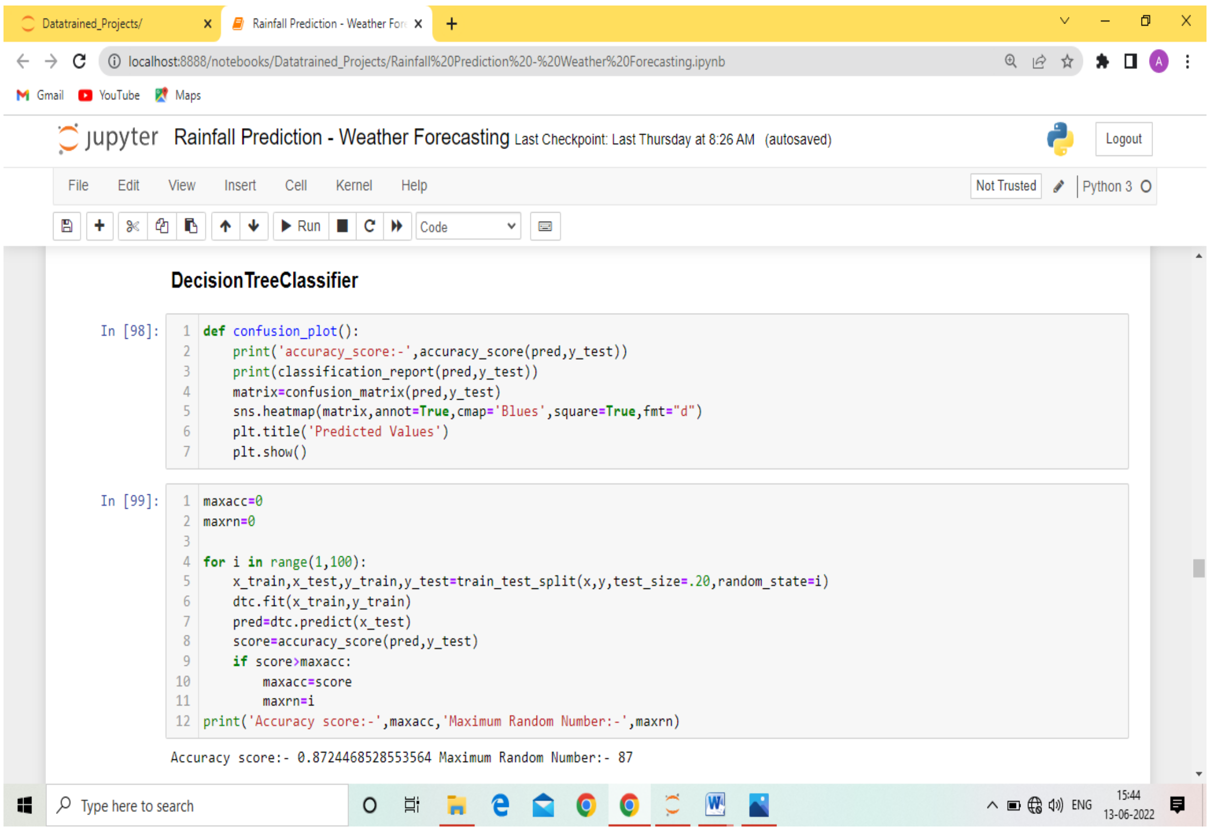
In [93]:
```
maxacc=0
maxrn=0

for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    lg.fit(x_train,y_train)
    pred=lg.predict(x_test)
    score=accuracy_score(pred,y_test)
    if score>maxacc:
        maxacc=score
        maxrn=i
print('Accuracy score:-',maxacc,'Maximum Random Number:-',maxrn)
```

```
Accuracy score:- 0.7969987494789496 Maximum Random Number:- 85
```

Fig 7:-Logistic Regression

Decision Tree have a natural if then else construction that makes it fit easily into a programmatic structure. They also are well suited to categorization problems where attributes or features are systematically checked to determine a final category. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables. This characteristics of Decision Tree makes it a good fit for our problem as our target variable is binary categorical variable.


Datatrained_Projects/
Rainfall Prediction - Weather For
localhost:8888/notebooks/Datatrained_Projects/Rainfall%20Prediction%20-%20Weather%20Forecasting.ipynb
Gmail
YouTube
Maps
jupyter
Rainfall Prediction - Weather Forecasting Last Checkpoint: Last Thursday at 8:26 AM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3
Run
Code
Type here to search
ENG
15:44
13-06-2022

DecisionTreeClassifier

```
In [98]:
def confusion_plot():
    print('accuracy_score:-',accuracy_score(pred,y_test))
    print(classification_report(pred,y_test))
    matrix=confusion_matrix(pred,y_test)
    sns.heatmap(matrix,annot=True,cmap='Blues',square=True,fmt="d")
    plt.title('Predicted Values')
    plt.show()
```

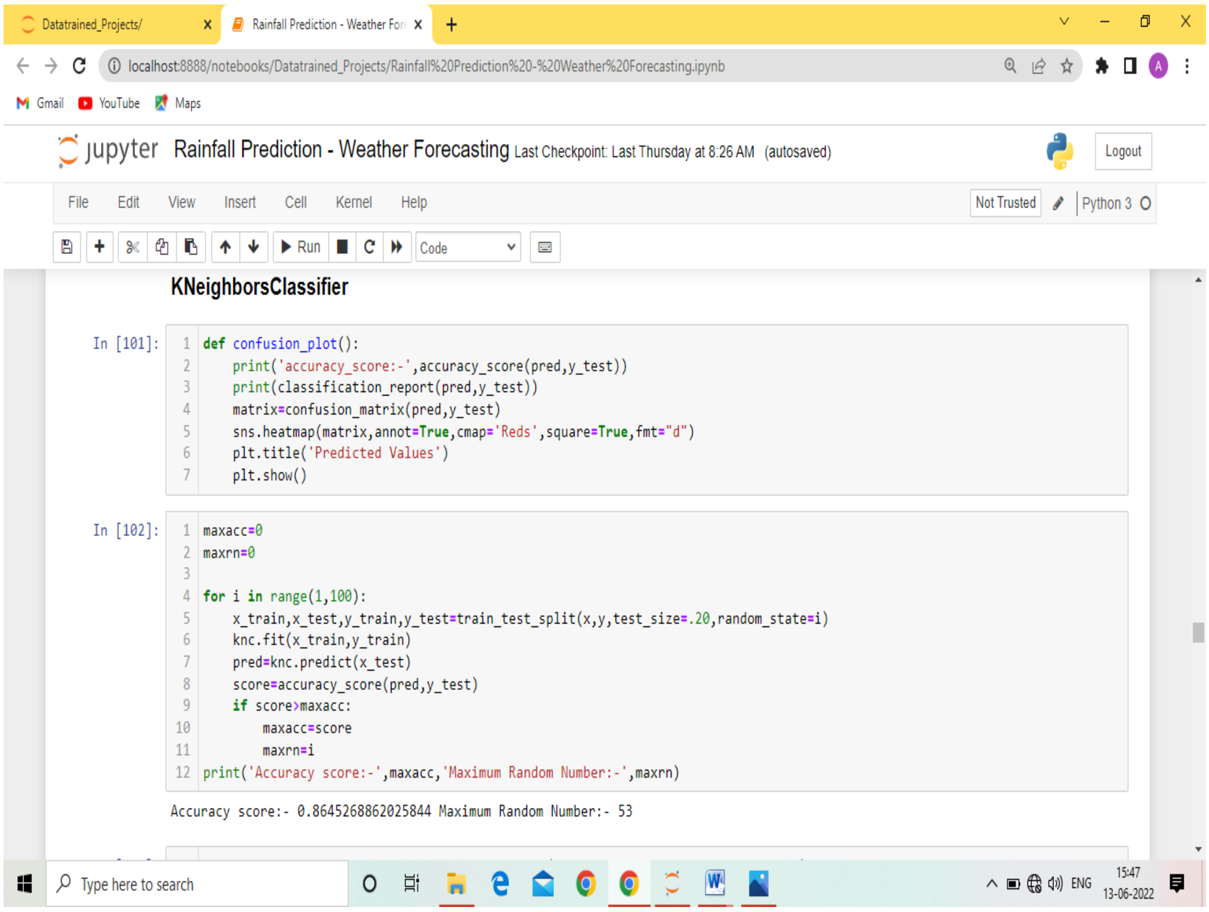
```
In [99]:
maxacc=0
maxrn=0

for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    dtc.fit(x_train,y_train)
    pred=dtc.predict(x_test)
    score=accuracy_score(pred,y_test)
    if score>maxacc:
        maxacc=score
        maxrn=i
print('Accuracy score:-',maxacc,'Maximum Random Number:-',maxrn)
```

```
Accuracy score:- 0.8724468528553564 Maximum Random Number:- 87
```

Fig 8:-Decision Tree Classifier

K - Nearest Neighbour is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure is determined from the dataset. Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. KNN performs better with a lower number of features than a large number of features. We can say that when the number of features increases than it requires more data. Increase in dimension also leads to the problem of overfitting. However, we have performed feature selection which helps to reduce dimension and hence KNN looks a good candidate for our problem.



Rainfall Prediction - Weather Forecasting Last Checkpoint: Last Thursday at 8:26 AM (autosaved)
KNeighborsClassifier

```
In [101]:
def confusion_plot():
    print('accuracy_score:-',accuracy_score(pred,y_test))
    print(classification_report(pred,y_test))
    matrix=confusion_matrix(pred,y_test)
    sns.heatmap(matrix,annot=True,cmap='Reds',square=True,fmt="d")
    plt.title('Predicted Values')
    plt.show()
```

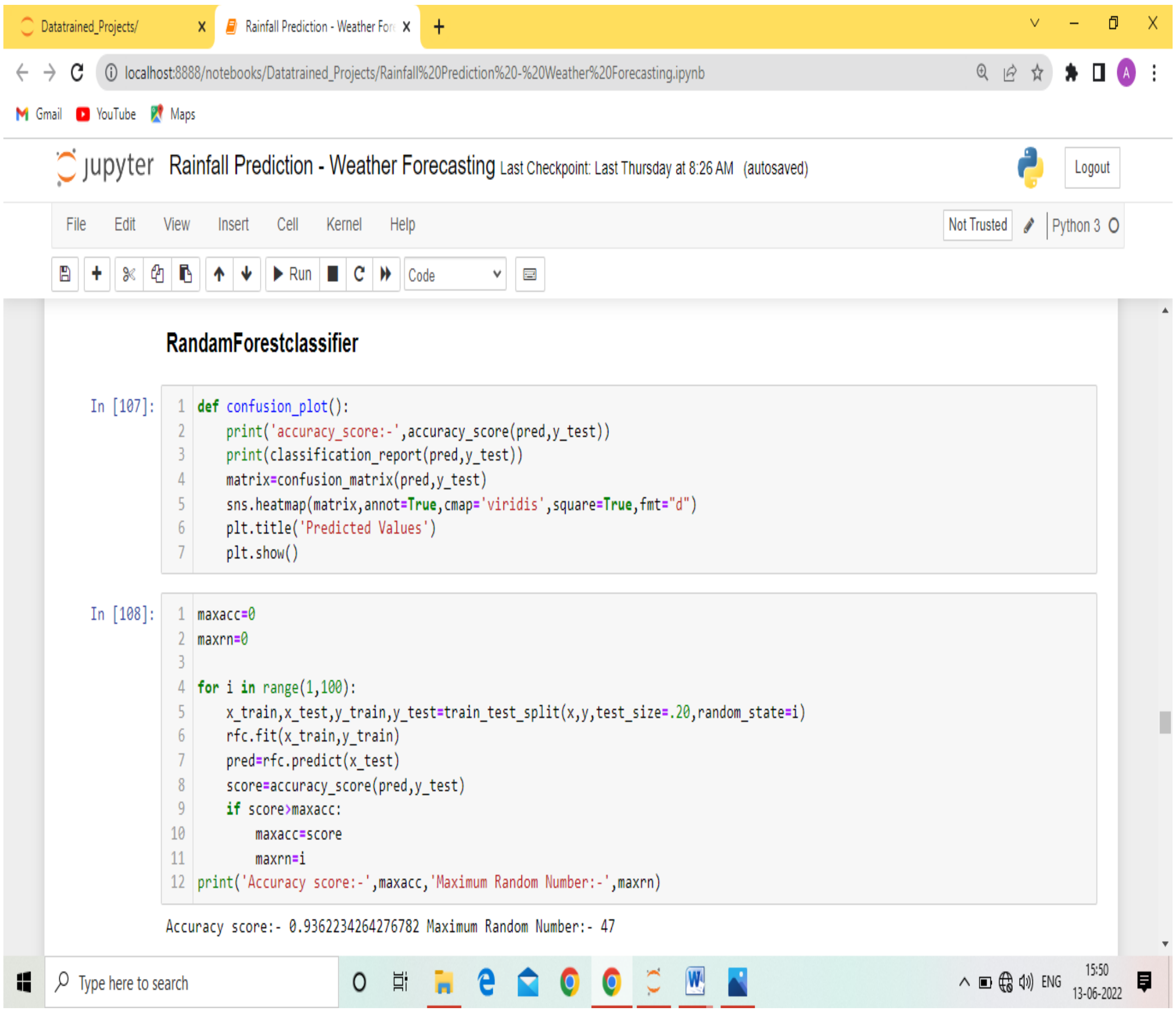
```
In [102]:
maxacc=0
maxrn=0

for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    knc.fit(x_train,y_train)
    pred=knc.predict(x_test)
    score=accuracy_score(pred,y_test)
    if score>maxacc:
        maxacc=score
        maxrn=i
print('Accuracy score:-',maxacc,'Maximum Random Number:-',maxrn)
```

```
Accuracy score:- 0.8645268862025844 Maximum Random Number:- 53
```

Fig 9: K-Neighbors Classifier

Random Forest is a supervised ensemble learning algorithm. Ensemble means that it takes a bunch of weak learners and have them work together to form one strong predictor. Here, we have a collection of decision trees, known as Forest. To classify a new object based on attributes, each tree gives a classification and we say the tree votes for that class. The forest chooses the classification having the most votes (over all the trees in the forest).
Our Model’s configuration: number of weak learners = 100, maximum depth of each tree = 4


Datatrained_Projects/
Rainfall Prediction - Weather For
localhost:8888/notebooks/Datatrained_Projects/Rainfall%20Prediction%20-%20Weather%20Forecasting.ipynb
Gmail
YouTube
Maps
jupyter
Rainfall Prediction - Weather Forecasting Last Checkpoint: Last Thursday at 8:26 AM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3
Run
Code
Type here to search
ENG
15:50
13-06-2022

RandamForestclassifier

In [107]:
```python
def confusion_plot():
    print('accuracy_score:-',accuracy_score(pred,y_test))
    print(classification_report(pred,y_test))
    matrix=confusion_matrix(pred,y_test)
    sns.heatmap(matrix,annot=True,cmap='viridis',square=True,fmt="d")
    plt.title('Predicted Values')
    plt.show()
```

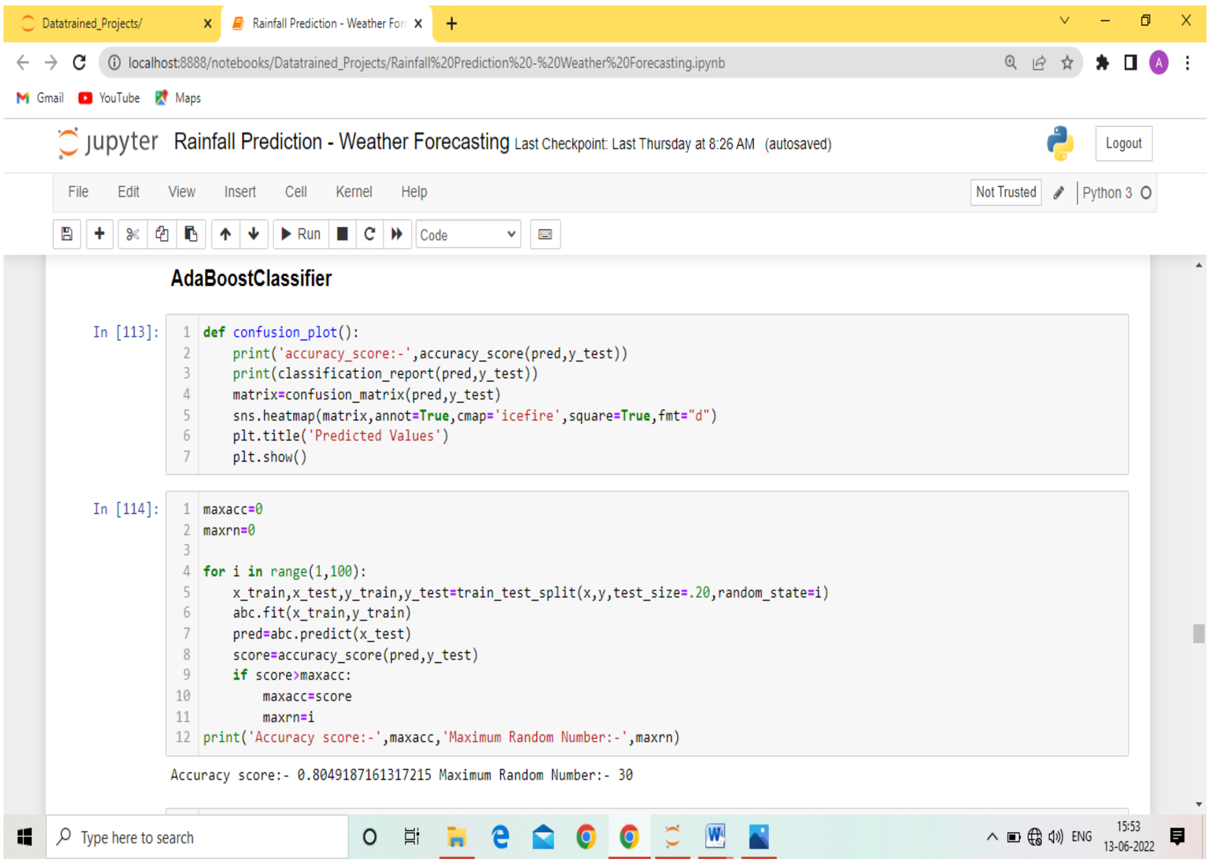
In [108]:
```python
maxacc=0
maxrn=0

for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    rfc.fit(x_train,y_train)
    pred=rfc.predict(x_test)
    score=accuracy_score(pred,y_test)
    if score>maxacc:
        maxacc=score
        maxrn=i
print('Accuracy score:-',maxacc,'Maximum Random Number:-',maxrn)
```

```
Accuracy score:- 0.9362234264276782 Maximum Random Number:- 47
```

Fig 10: Random Forest Classifier

AdaBoost fits a sequence of weak learners on different weighted training data. It starts by predicting original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observations which have been predicted incorrectly. Being an iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy.
Our Model's configuration: number of weak learners = 50


Datatrained_Projects/
Rainfall Prediction - Weather For
localhost:8888/notebooks/Datatrained_Projects/Rainfall%20Prediction%20-%20Weather%20Forecasting.ipynb
Gmail
YouTube
Maps
jupyter Rainfall Prediction - Weather Forecasting Last Checkpoint: Last Thursday at 8:26 AM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3
Run
Code
Type here to search
ENG
15:53
13-06-2022

AdaBoostClassifier

```
In [113]: def confusion_plot():
              print('accuracy_score:-',accuracy_score(pred,y_test))
              print(classification_report(pred,y_test))
              matrix=confusion_matrix(pred,y_test)
              sns.heatmap(matrix,annot=True,cmap='icefire',square=True,fmt="d")
              plt.title('Predicted Values')
              plt.show()
```

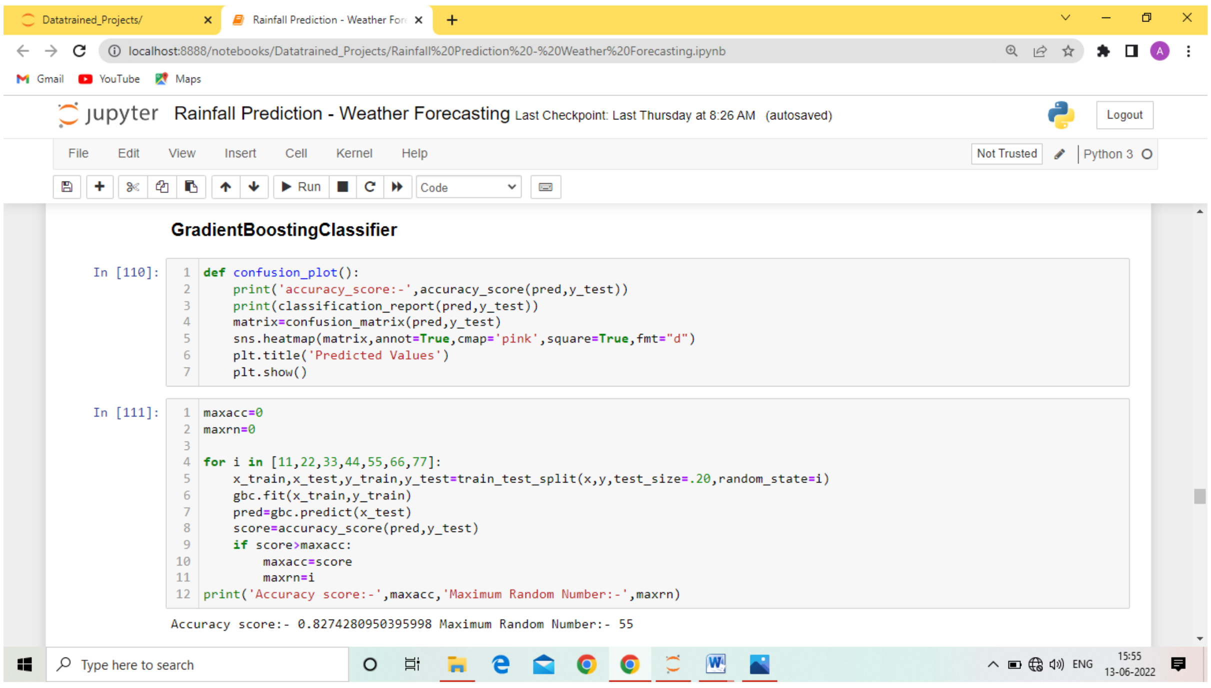
```
In [114]: maxacc=0
          maxrn=0

          for i in range(1,100):
              x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
              abc.fit(x_train,y_train)
              pred=abc.predict(x_test)
              score=accuracy_score(pred,y_test)
              if score>maxacc:
                  maxacc=score
                  maxrn=i
          print('Accuracy score:-',maxacc,'Maximum Random Number:-',maxrn)
```

```
Accuracy score:- 0.8049187161317215 Maximum Random Number:- 30
```

Fig 11: Ada Boost Classifier

Gradient Boosting Here, many models are trained sequentially. Each new model gradually minimizes the loss function (y = ax + b + e, where e is the error term) of the whole system using Gradient Descent method. The learning method consecutively fits new models to give a more accurate estimate of the response variable. The main idea behind this algorithm is to construct new base learners which can be optimally correlated with negative gradient of the loss function, relevant to the whole ensemble.



Rainfall Prediction - Weather Forecasting Last Checkpoint: Last Thursday at 8:26 AM (autosaved)

GradientBoostingClassifier

In [110]:
```python
def confusion_plot():
    print('accuracy_score:-',accuracy_score(pred,y_test))
    print(classification_report(pred,y_test))
    matrix=confusion_matrix(pred,y_test)
    sns.heatmap(matrix,annot=True,cmap='pink',square=True,fmt="d")
    plt.title('Predicted Values')
    plt.show()
```

In [111]:
```python
maxacc=0
maxrn=0

for i in [11,22,33,44,55,66,77]:
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    gbc.fit(x_train,y_train)
    pred=gbc.predict(x_test)
    score=accuracy_score(pred,y_test)
    if score>maxacc:
        maxacc=score
        maxrn=i
print('Accuracy score:-',maxacc,'Maximum Random Number:-',maxrn)
```

```
Accuracy score:- 0.8274280950395998 Maximum Random Number:- 55
```

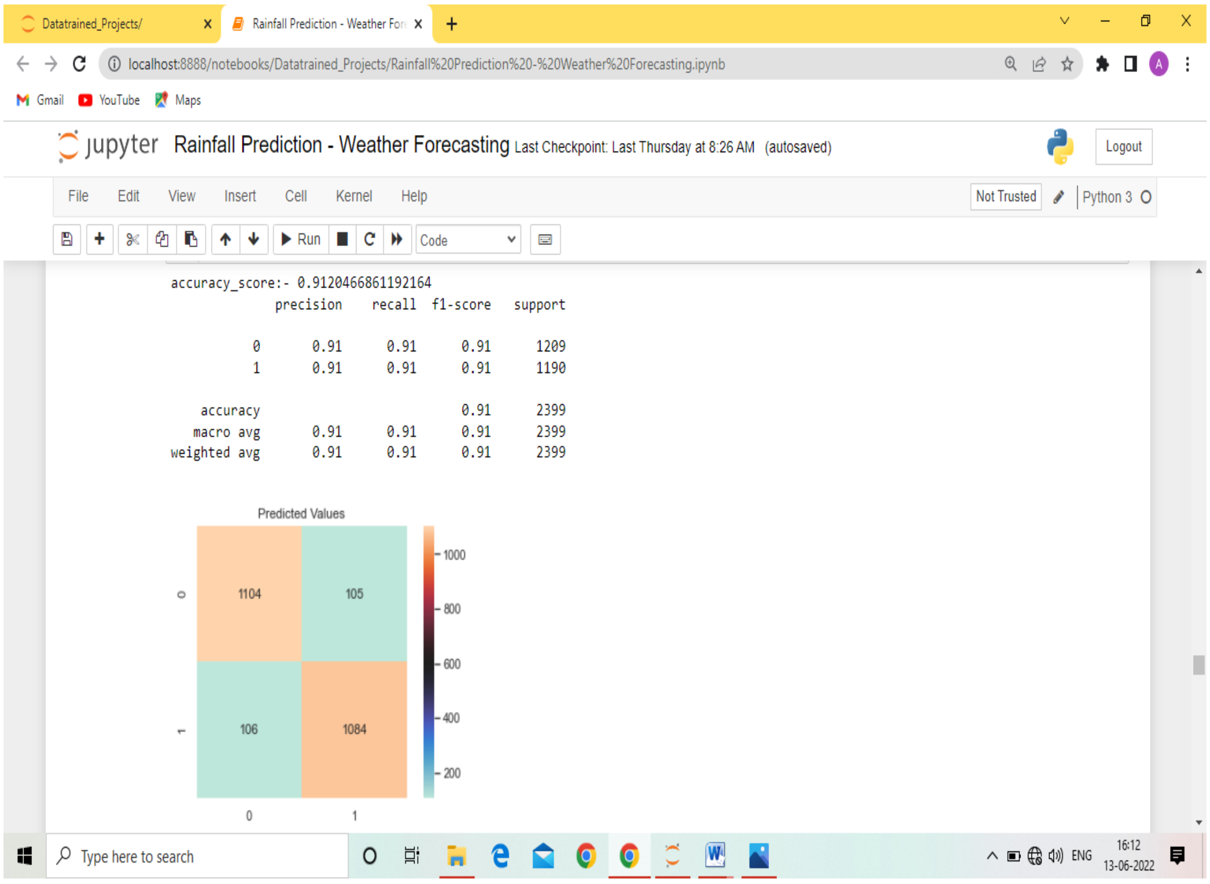
Fig 12: Gradient Boosting Classifier

4. Evaluation

For evaluating our classifiers we used below evaluation metrics .

Accuracy is the ratio of number of correct predictions to the total number of input samples. It works well only if there are equal number of samples belonging to each class. As we have, balanced data, we will also consider other metrics like classification report and confusion matrix.

Random Forest Classifier is the best model for this Dataset


Datatrained_Projects/
Rainfall Prediction - Weather For
localhost:8888/notebooks/Datatrained_Projects/Rainfall%20Prediction%20-%20Weather%20Forecasting.ipynb
Gmail
YouTube
Maps
jupyter
Rainfall Prediction - Weather Forecasting Last Checkpoint: Last Thursday at 8:26 AM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3
Run
Code
Predicted Values
1104
105
106
1084
0
1
1000
800
600
400
200
Type here to search
ENG
16:12
13-06-2022

```
accuracy_score:- 0.9120466861192164
              precision    recall  f1-score   support

           0       0.91      0.91      0.91      1209
           1       0.91      0.91      0.91      1190

    accuracy                           0.91      2399
   macro avg       0.91      0.91      0.91      2399
weighted avg       0.91      0.91      0.91      2399
```

Fig 13: Random Forest Classifier Accuracy

Confusion Matrix gives us a matrix as output and describes the complete performance of the model. It focuses on True Positives - the cases in which we predicted YES and the actual output was also YES; True Negatives - the cases in which we predicted NO and the actual output was NO; False Positives the cases in which we predicted YES and the actual output was NO; False Negatives - the cases in which we predicted NO and the actual output was YES.

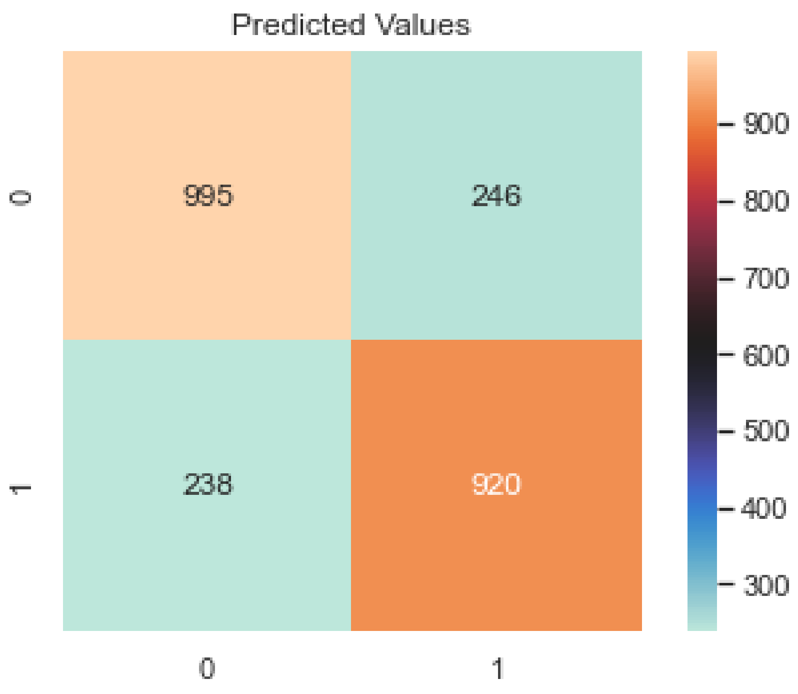
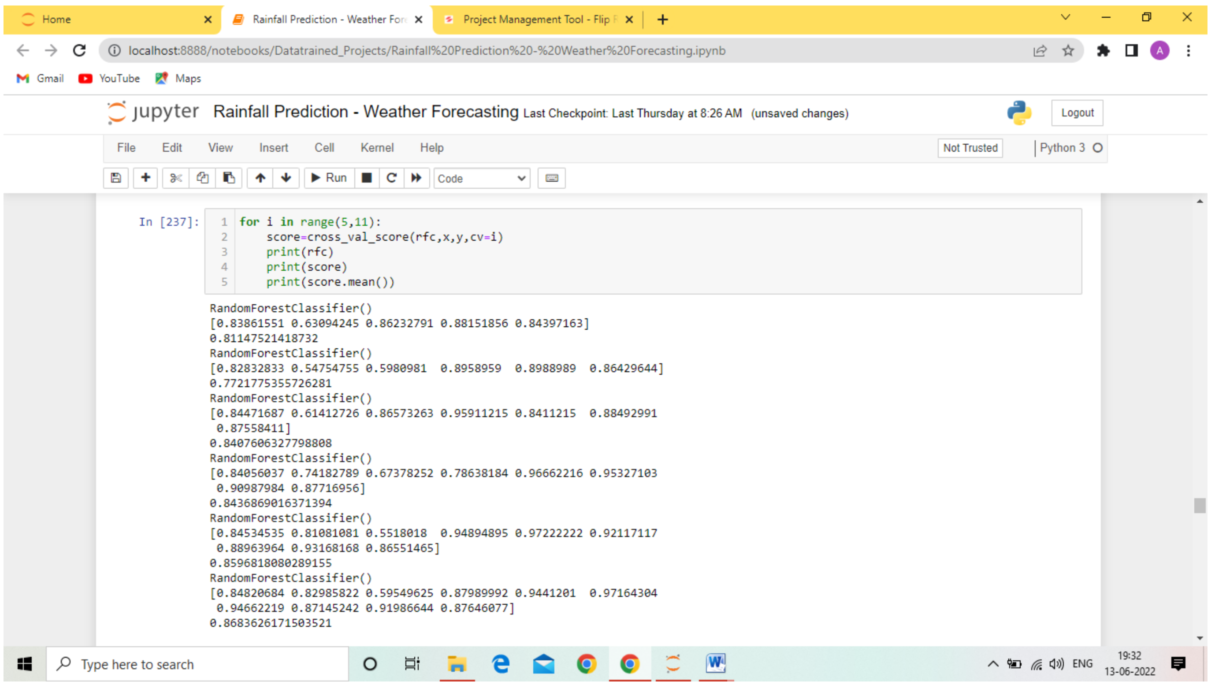


Fig 14: Confusion Matrix

Stratified Cross Validation: As our data is imbalanced, we trained our models using a stratified Cross validation approach where the data is divided into 5 folds each of equal proportion of positives and negatives. These metrics would be more reliable and less biased. We used the value of CV as 10.


Home
Rainfall Prediction - Weather For
Project Management Tool - Flip
localhost:8888/notebooks/Datatrained_Projects/Rainfall%20Prediction%20-%20Weather%20Forecasting.ipynb
Gmail
YouTube
Maps
jupyter
Rainfall Prediction - Weather Forecasting Last Checkpoint: Last Thursday at 8:26 AM (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3
Run
Code
Type here to search
ENG
19:32
13-06-2022

```
In [237]:
for i in range(5,11):
    score=cross_val_score(rfc,x,y,cv=i)
    print(rfc)
    print(score)
    print(score.mean())
```

```
RandomForestClassifier()
[0.83861551 0.63094245 0.86232791 0.88151856 0.84397163]
0.81147521418732
RandomForestClassifier()
[0.82832833 0.54754755 0.5980981  0.8958959  0.8988989  0.86429644]
0.7721775355726281
RandomForestClassifier()
[0.84471687 0.61412726 0.86573263 0.95911215 0.8411215  0.88492991
 0.87558411]
0.8407606327798808
RandomForestClassifier()
[0.84056037 0.74182789 0.67378252 0.78638184 0.96662216 0.95327103
 0.90987984 0.87716956]
0.8436869016371394
RandomForestClassifier()
[0.84534535 0.81081081 0.5518018  0.94894895 0.97222222 0.92117117
 0.88963964 0.93168168 0.86551465]
0.8596818080289155
RandomForestClassifier()
[0.84820684 0.82985822 0.59549625 0.87989992 0.9441201  0.97164304
 0.94662219 0.87145242 0.91986644 0.87646077]
0.8683626171503521
```

Fig 15: Cross Validation

Experiments and Results

For all the experiments and development of classifiers, we used Python 3 and Google colab’s Jupyter Notebook. We used libraries such as Sckit Learn, Matplotlib, Seaborn, Pandas, Numpy and Imblearn. We used jupyter for implementing Decision Table.
We carried experiments with different input data; one with the original dataset, then with the undersampled dataset and last one with the oversam- pled dataset. We splitted out dataset in ratio of 80:20 for training and testing purpose.

We get Best Result from the Random Forest Classifier with 91.20% Accuracy.

Conclusion

In this paper, we explored and applied several preprocessing steps and learned there impact on the overall performance of our classifiers. We also carried a comparative study of all the classifiers with different input data and observed how the input data can affect the model predictions.
Humidity 3pm,cloud 3pm,Rain Today and Rainfall is most important variables for predict that RainTomorrow (Target) variable.