# LP1 - Data Analytics Mini project

## Problem Statement:-

To classify air quality of a particular region based on the given parameters.

## Team Members :

41250 - Sahil Patil
41254 - Pralhad Kulkarni
41258 - Srusti Raybhoge

## Abstract:-

The main motive of the project is to classify the air quality of the input parameters by creating a model with the help of a given dataset . The attributes of the dataset include the amount of various elements present in the air at that very moment. We have used different ML algorithms to solve the same problem and create a model

## H/W & S/W requirements:-

Python, Python libraries, Jupyter Notebook/Google Colab/ Python IDE.

## Objective:-

1. To understand classification models
2. To predict air quality on the given inputs
3. Understand data preprocessing

## Scope:-

1. To predict the air quality based on the given factors, place and time.
2. Study two classifiers

## Introduction:-

**Classification:** refers to a predictive modeling problem where a class label is predicted for a given example of input data.

**Types of Classification:**

1. On the basis of Labels:
    a. Binary labelled
    b. Multi labelled

2. Linear Models:
    a. Logistic Regression
    b. Support Vector Machines

3. Nonlinear Models

    a. K-nearest Neighbors (KNN)
    b. Kernel Support Vector Machines (SVM)
    c. Naïve Bayes
    d. Decision Tree Classification
    e. Random Forest Classification

**Data Preprocessing Steps:**

1. Remove empty columns
2. Fill in empty cells with appropriate values ( here median of the values in that particular column )
3. Convert the string type data into equivalent numeric data.

**SMOT (Synthetic Minority Oversampling Technique):**

Sometimes the dataset may be imbalanced i.e. some classes have a majority number of samples than other classes. This may affect the model and we get poor performance with minority class.

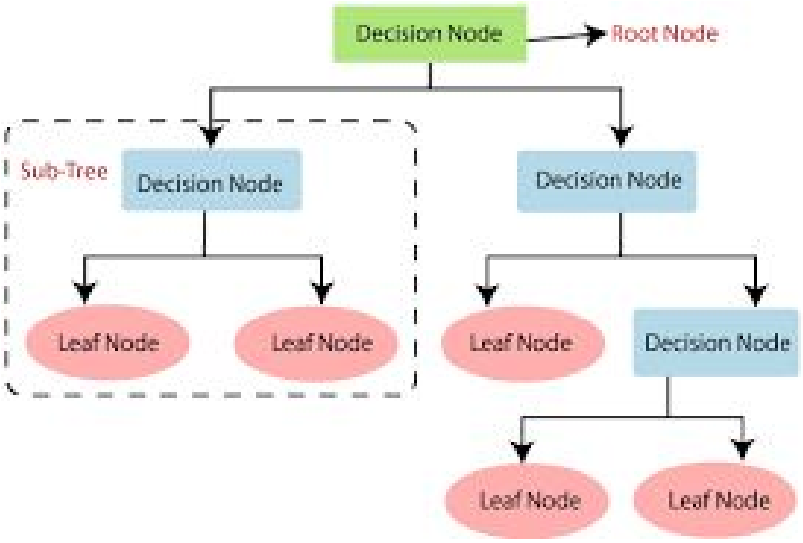
To avoid this we can do oversampling or undersampling of data. An improvement on duplicating examples from the minority class is to synthesize new examples from the minority class. This is a type of data augmentation for tabular data and can be very effective.

We have used random forest and decision tree classification to classify the dataset into appropriate air quality class and predict the class for the input.

**Decision Tree:**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

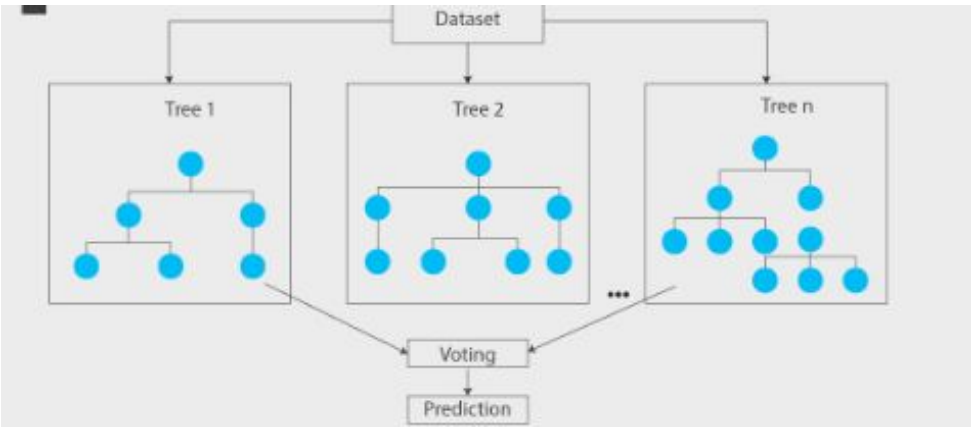
In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node **.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.




**Random Forest:**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

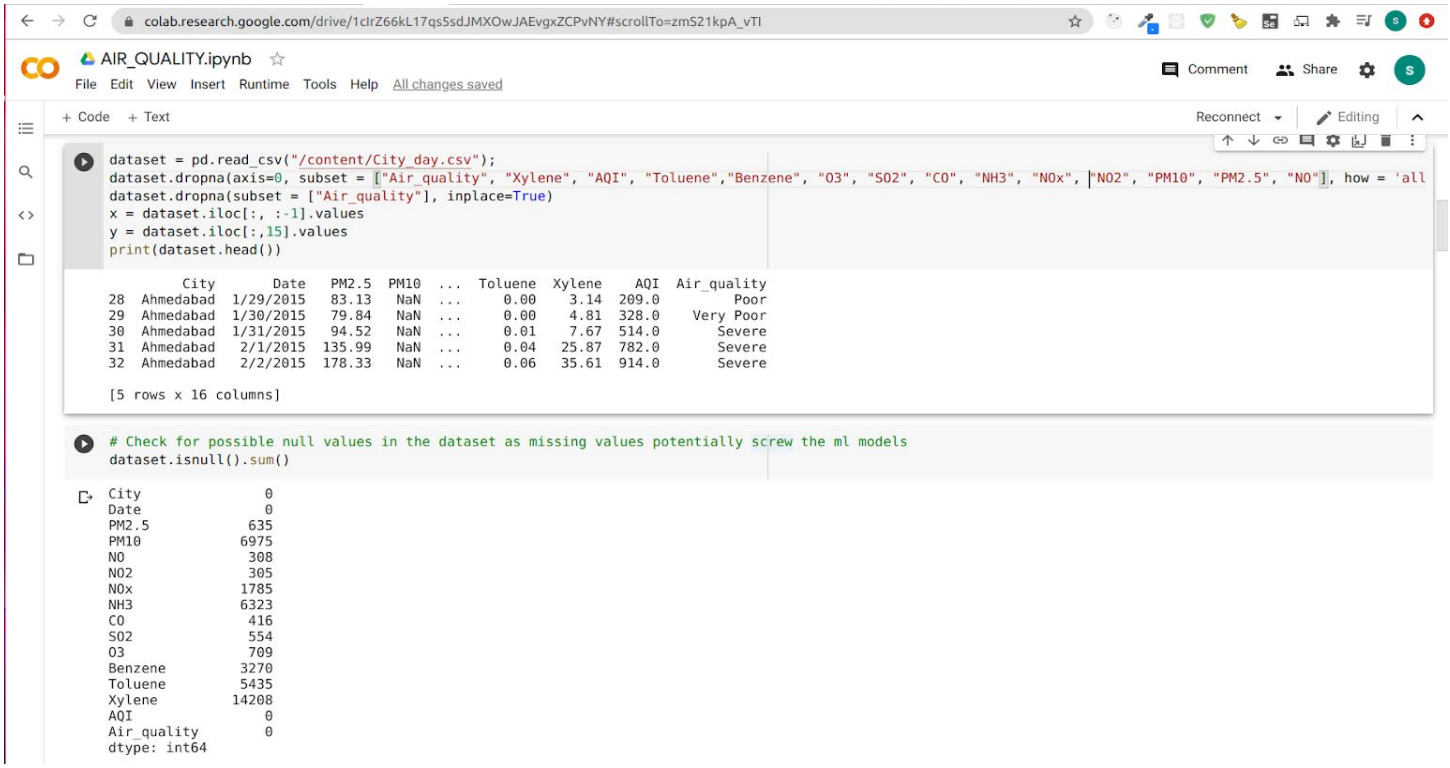
As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

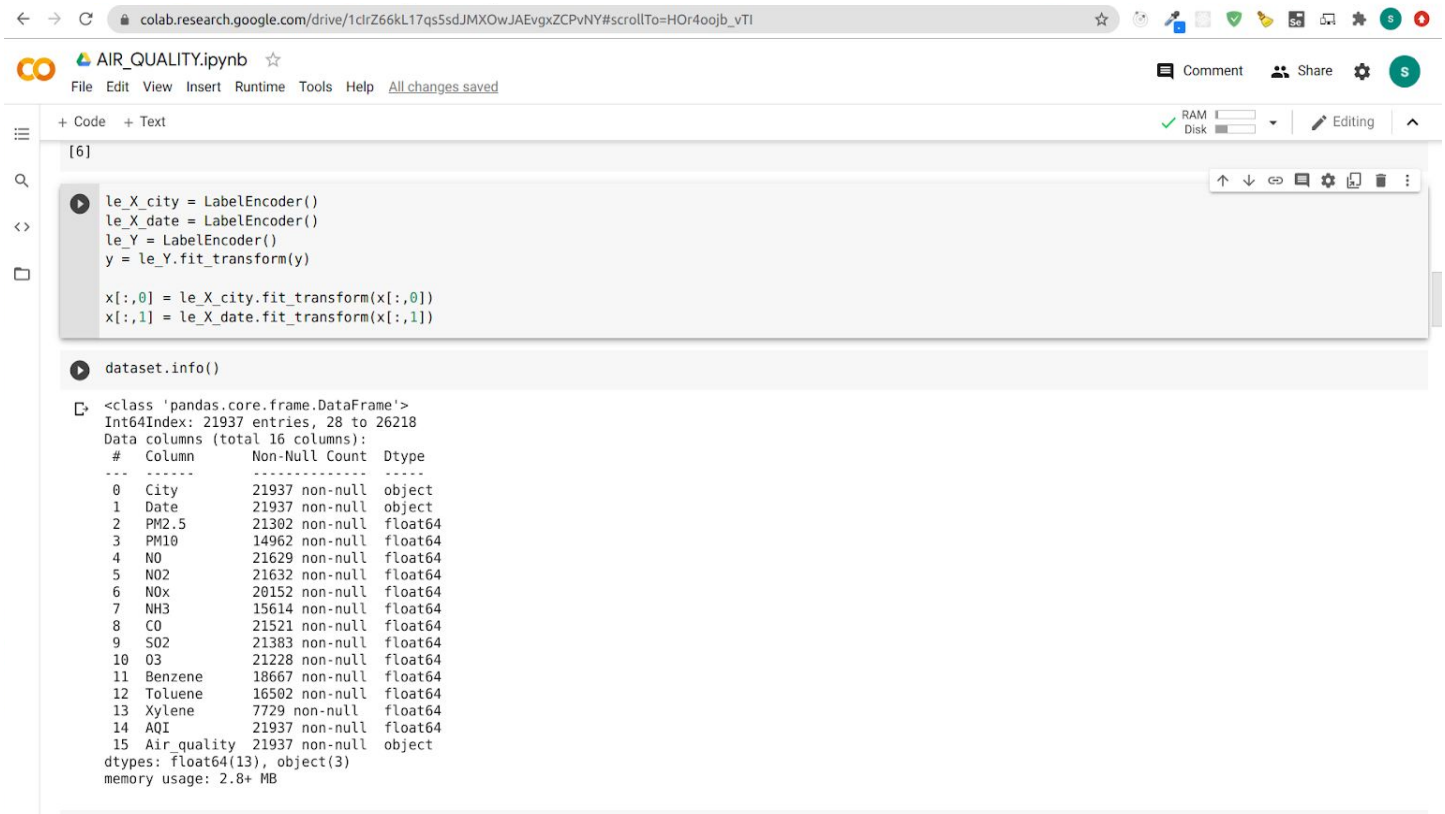



**Features from the dataset:**
City,Date,PM2.5,PM10,NO,NO2,NOx,NH3,CO,SO2,O3,Benzene,Toluene,Xylene,AQI,Air_quality

**Results:-**

Dataset overview:




Mean, Max, Min, Standard deviation, Count of each column

colab.research.google.com/drive/1clrZ66kL17qs5sdJMXOwJAEvgxZCPvNY#scrollTo=HOr4oojb_vTI

AIR_QUALITY.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

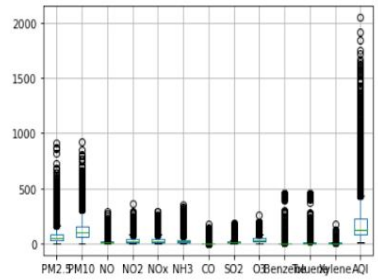
+ Code + Text

RAM Disk Editing

| | PM2.5 | PM10 | NO | NO2 | NOx | NH3 | CO | SO2 | O3 | Benzene | Toluene | Xylene | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 21302.000000 | 14962.000000 | 21629.000000 | 21632.000000 | 20152.000000 | 15614.000000 | 21521.000000 | 21383.000000 | 21228.000000 | 18667.000000 | 16502.000000 | 7729.000000 | 21937.0 |
| mean | 71.819536 | 125.867953 | 18.336627 | 29.537978 | 33.470678 | 25.751801 | 2.552130 | 14.801633 | 34.833688 | 3.465744 | 9.910826 | 3.741817 | 175.8 |
| std | 65.402215 | 93.019867 | 22.987785 | 25.341563 | 31.934069 | 27.323008 | 7.508587 | 18.342644 | 21.432535 | 16.922770 | 22.126794 | 7.134129 | 146.1 |
| min | 0.040000 | 0.030000 | 0.030000 | 0.010000 | 0.000000 | 0.010000 | 0.000000 | 0.010000 | 0.100000 | 0.000000 | 0.000000 | 0.000000 | 13.0 |
| 25% | 31.670000 | 61.692500 | 5.950000 | 12.100000 | 13.147500 | 9.630000 | 0.610000 | 5.610000 | 19.437500 | 0.230000 | 1.060000 | 0.300000 | 85.0 |
| 50% | 52.430000 | 101.555000 | 10.170000 | 22.360000 | 24.310000 | 18.420000 | 0.970000 | 9.170000 | 31.330000 | 1.180000 | 3.370000 | 1.270000 | 124.0 |
| 75% | 85.987500 | 158.837500 | 20.910000 | 38.702500 | 42.090000 | 33.130000 | 1.560000 | 15.500000 | 45.862500 | 3.100000 | 10.047500 | 4.350000 | 226.0 |
| max | 914.940000 | 917.080000 | 287.140000 | 362.210000 | 293.100000 | 352.890000 | 175.810000 | 186.080000 | 257.730000 | 455.030000 | 454.850000 | 170.370000 | 2049.0 |

```
print(dataset.boxplot())
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```




colab.research.google.com/drive/1clrZ66kL17qs5sdJMXOwJAEvgxZCPvNY#scrollTo=HOr4oojb_vTI

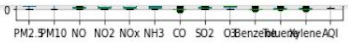
AIR_QUALITY.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

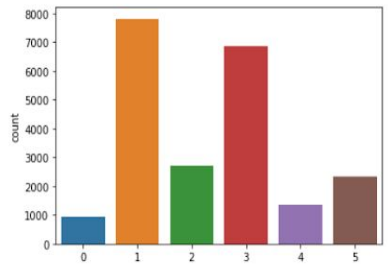
RAM Disk Editing




```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 0)
```

```
ax = sns.countplot(y)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp
```




```
#Before OverSampling
ax = sns.countplot(y_train)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp
```

colab.research.google.com/drive/1clrZ66kL17qs5sdJMXOwJAEvgxZCPvNY#scrollTo=HOr4oojb_vTI

AIR_QUALITY.ipynb

File Edit View Insert Runtime Tools Help All changes saved

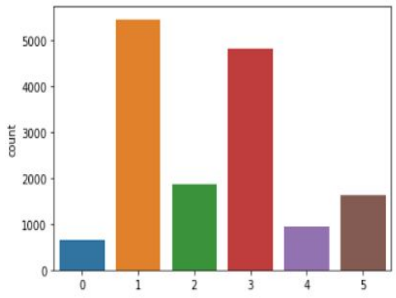
Comment Share

+ Code + Text

RAM Disk Editing

```
[ ] #Before OverSampling
    ax = sns.countplot(y_train)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp
```




```
print('Classes and number of values in trainset',Counter(y_train))
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
x_train, y_train = oversample.fit_resample(x_train,y_train)
print('Classes and number of values in trainset after SMOTE:',Counter(y_train))
med=np.median(x_train,axis=0)
```

```
Classes and number of values in trainset Counter({1: 5450, 3: 4808, 2: 1867, 5: 1618, 4: 951, 0: 661})
/usr/local/lib/python3.6/dist-packages/sklearn/externals/six.py:31: FutureWarning:
```

colab.research.google.com/drive/1clrZ66kL17qs5sdJMXOwJAEvgxZCPvNY#scrollTo=HOr4oojb_vTI

AIR_QUALITY.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

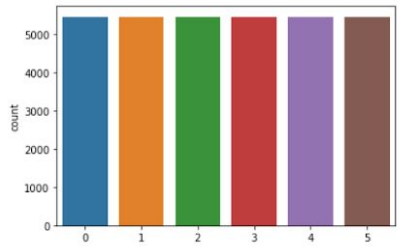
+ Code + Text

RAM Disk Editing

```
[ ] Function safe_indexing is deprecated; safe_indexing is deprecated in version 0.22 and will be removed in version 0.24.

    Classes and number of values in trainset after SMOTE: Counter({3: 5450, 1: 5450, 2: 5450, 5: 5450, 4: 5450, 0: 5450})
    /usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning:

    Function safe_indexing is deprecated; safe_indexing is deprecated in version 0.22 and will be removed in version 0.24.

    /usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning:

    Function safe_indexing is deprecated; safe_indexing is deprecated in version 0.22 and will be removed in version 0.24.
```

```
[ ] #After Oversampling
    sns.countplot(y_train)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp

<matplotlib.axes._subplots.AxesSubplot at 0x7fc687c26e10>
```



colab.research.google.com/drive/1clrZ66kL17qs5sdJMXOwJAEvgxZCPvNY#scrollTo=CrvARdwM_vTK

AIR_QUALITY.ipynb

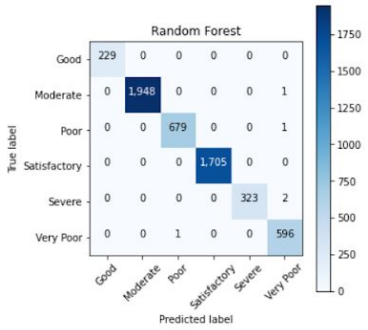
File Edit View Insert Runtime Tools Help All changes saved

Comment Share S

+ Code + Text Reconnect Editing

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm,title="Random Forest")

a = accuracy_score(y_test,y_pred)
precision = precision_score(y_test,y_pred, average='micro')
recall = recall_score(y_test,y_pred, average='micro')
f1 = f1_score(y_test,y_pred, average='micro')
print("The accuracy of this model is : ", a*100)
print("Precision : ",precision*100)
print("Recall : ",recall*100)
print("F1 Score : ",f1*100)
```

```
The accuracy of this model is :  99.90884229717412
Precision :  99.90884229717412
Recall :  99.90884229717412
F1 Score :  99.90884229717412
```




colab.research.google.com/drive/1clrZ66kL17qs5sdJMXOwJAEvgxZCPvNY#scrollTo=CrvARdwM_vTK

AIR_QUALITY.ipynb

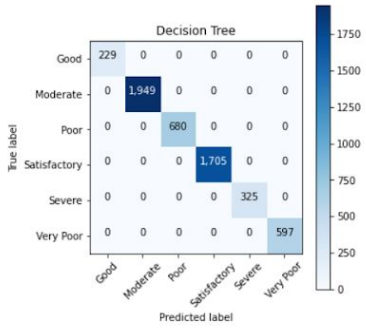
File Edit View Insert Runtime Tools Help All changes saved

Comment Share S

+ Code + Text Reconnect Editing

```
[ ] cm = confusion_matrix(y_test, y_pred)
    plot_confusion_matrix(cm,title="Decision Tree")

    a = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred, average='micro')
    recall = recall_score(y_test,y_pred, average='micro')
    f1 = f1_score(y_test,y_pred, average='micro')
    print("The accuracy of this model is : ", a*100)
    print("Precision : ",precision*100)
    print("Recall : ",recall*100)
    print("F1 Score : ",f1*100)
```

```
The accuracy of this model is :  100.0
Precision :  100.0
Recall :  100.0
F1 Score :  100.0
```



## TEST CASES:

```
[16] res = []
     city = le_X_city.fit_transform(["Pune"])
     date = le_X_date.fit_transform(["12/12/2020"])
     ls = [city[0] ,date[0], 83.13, 101.555, 6.93, 28.71, 33.72, 18.42, 6.93, 49.52, 59.76, 0.02, 0.0, 3.14, 209.0]
     lst = []
     lst.append(ls)
     temp = le_Y.inverse_transform(clf.predict(lst))
     temp = temp.tolist()
     res.append(temp[0])

     temp = le_Y.inverse_transform(dt.predict(lst))
     temp = temp.tolist()
     res.append(temp[0])

     print("Random Forest : ",res[0])
     print("Decision Tree : ",res[1])

     Random Forest :  Poor
     Decision Tree :  Poor
```

```
res = []
city = le_X_city.fit_transform(["Patna"])
date = le_X_date.fit_transform(["12/12/2020"])
ls = [city[0] ,date[0],49.34,0,20.33,26.7,21.5,0,0.91,49.93,49.47,1.43,16.43,5.49,94]
lst = []
lst.append(ls)
temp = le_Y.inverse_transform(clf.predict(lst))
temp = temp.tolist()
res.append(temp[0])

temp = le_Y.inverse_transform(dt.predict(lst))
temp = temp.tolist()
res.append(temp[0])
```

```
res = []
city = le_X_city.fit_transform(["Patna"])
date = le_X_date.fit_transform(["12/12/2020"])
ls = [city[0] ,date[0],49.34,0,20.33,26.7,21.5,0,0.91,49.93,49.47,1.43,16.43,5.49,94]
lst = []
lst.append(ls)
temp = le_Y.inverse_transform(clf.predict(lst))
temp = temp.tolist()
res.append(temp[0])

temp = le_Y.inverse_transform(dt.predict(lst))
temp = temp.tolist()
res.append(temp[0])

print("Random Forest : ",res[0])
print("Decision Tree : ",res[1])

Random Forest :  Satisfactory
Decision Tree :  Satisfactory
```

## Conclusion:-

Thus we studied different classification algorithms and used two of them to predict the quality of air in a particular region at a particular time.