importing libraries

In [59]:

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

importing dataset

In [60]:

```
dataset = pd.read_csv('weatherAUS.csv')
X = dataset.iloc[:,[1,2,3,4,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]].values
Y = dataset.iloc[:,-1].values
```

In [61]:

```
print(X)
```

```
[['Albury' 13.4 22.9 ... 16.9 21.8 'No']
 ['Albury' 7.4 25.1 ... 17.2 24.3 'No']
 ['Albury' 12.9 25.7 ... 21.0 23.2 'No']
 ...
 ['Uluru' 5.4 26.9 ... 12.5 26.1 'No']
 ['Uluru' 7.8 27.0 ... 15.1 26.0 'No']
 ['Uluru' 14.9 nan ... 15.0 20.9 'No']]
```

In [62]:

```
print(Y)
```

```
['No' 'No' 'No' ... 'No' 'No' nan]
```

In [63]:

```
# 1D list to 2D list
Y = Y.reshape(-1,1)
```

In [64]:

```
print(Y)
```

```
[['No']
 ['No']
 ['No']
 ...
 ['No']
 ['No']
 [nan]]
```

dealing with invalid data

In [65]:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan,strategy='most_frequent')
X = imputer.fit_transform(X)
Y = imputer.fit_transform(Y)
```

In [66]:

```
print(X)
```

```
[['Albury' 13.4 22.9 ... 16.9 21.8 'No']
 ['Albury' 7.4 25.1 ... 17.2 24.3 'No']
 ['Albury' 12.9 25.7 ... 21.0 23.2 'No']
 ...
 ['Uluru' 5.4 26.9 ... 12.5 26.1 'No']
 ['Uluru' 7.8 27.0 ... 15.1 26.0 'No']
 ['Uluru' 14.9 20.0 ... 15.0 20.9 'No']]
```

In [67]:

```
print(Y)
```

```
[['No']
 ['No']
 ['No']
 ...
 ['No']
 ['No']
 ['No']]
```

Encoding Dataset

In [68]:

```
from sklearn.preprocessing import LabelEncoder
le1 = LabelEncoder()
X[:,0] = le1.fit_transform(X[:,0])
le2 = LabelEncoder()
X[:,4] = le2.fit_transform(X[:,4])
le3 = LabelEncoder()
X[:,6] = le3.fit_transform(X[:,6])
le4 = LabelEncoder()
X[:,7] = le4.fit_transform(X[:,7])
le5 = LabelEncoder()
X[:,-1] = le5.fit_transform(X[:,-1])
le6 = LabelEncoder()
Y = le6.fit_transform(Y)
```

```
C:\New folder\lib\site-packages\sklearn\preprocessing\_label.py:115: DataConversionWarni
ng: A column-vector y was passed when a 1d array was expected. Please change the shape o
f y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [69]:

```
print(X)
```

```
[[2 13.4 22.9 ... 16.9 21.8 0]
 [2 7.4 25.1 ... 17.2 24.3 0]
 [2 12.9 25.7 ... 21.0 23.2 0]
 ...
 [41 5.4 26.9 ... 12.5 26.1 0]
 [41 7.8 27.0 ... 15.1 26.0 0]
 [41 14.9 20.0 ... 15.0 20.9 0]]
```

In [70]:

```
print(Y)
```

```
[0 0 0 ... 0 0 0]
```

feature scaling

In [71]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

In [72]:

```
print(X)
```

```
[[-1.53166617  0.19132753 -0.04135977 ... -0.01407077  0.02310362
  -0.52979545]
 [-1.53166617 -0.75105231  0.26874452 ...  0.03244663  0.387799
  -0.52979545]
 [-1.53166617  0.11279588  0.35331842 ...  0.62166712  0.22733303
  -0.52979545]
 ...
 [ 1.20928479 -1.06517892  0.52246622 ... -0.69632607  0.65037966
  -0.52979545]
 [ 1.20928479 -0.68822699  0.53656187 ... -0.29317521  0.63579185
```

```
  -0.52979545]
 [ 1.20928479  0.42692249 -0.45013361 ... -0.30868102 -0.10818671
  -0.52979545]]
```

splitting Dataset into Training set and Test set

In [73]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

In [74]:
```
print(X_train)
```

```
[[ 0.22535368  1.03946939  0.07140543 ...  0.68369032  0.08145488
  -0.52979545]
 [ 1.42012717 -0.45263203  0.11369237 ... -0.41722163  0.22733303
  -0.52979545]
 [ 0.50647685 -0.20133073 -0.14002932 ... -0.06058818 -0.02065982
   1.88752093]
 ...
 [ 1.0687232   0.75675544  0.93124006 ...  1.10234698  1.07342629
  -0.52979545]
 [ 0.57675765 -0.04426743 -0.16822062 ...  0.01694083 -0.28324049
   1.88752093]
 [ 1.63096955 -0.0285611  -0.91529006 ... -0.35519842 -0.76463838
  -0.52979545]]
```

In [75]:
```
print(Y_train)
```

```
[1 0 0 ... 0 0 0]
```

Training Model

In [76]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=150,random_state=0)
classifier.fit(X_train,Y_train)
```

Out[76]:
```
RandomForestClassifier(n_estimators=150, random_state=0)
```

In [77]:
```
classifier.score(X_train,Y_train)
```

Out[77]:
```
0.9999398460057748
```

In [78]:
```
y_pred = classifier.predict(X_test)
```

In [79]:
```
print(y_pred)
```

```
[0 0 0 ... 0 0 0]
```

In [80]:
```
y_pred = le6.inverse_transform(y_pred)
```

In [81]:
```
print(y_pred)
```

```
['No' 'No' 'No' ... 'No' 'No' 'No']
```

In [82]:
```
print(Y_test)
```

```
[1 1 0 ... 1 0 0]
```

In [83]:
```
Y_test = le6.inverse_transform(Y_test)
```

```
In [84]: print(Y_test)
```

```
['Yes' 'Yes' 'No' ... 'Yes' 'No' 'No']
```

```
In [85]: Y_test = Y_test.reshape(-1,1)
         y_pred = y_pred.reshape(-1,1)
```

```
In [86]: df = np.concatenate((Y_test,y_pred),axis=1)
         dataframe = pd.DataFrame(df,columns=['Rain on Tomorrow','Prediction of Rain'])
```

```
In [87]: print(df)
```

```
[['Yes' 'No']
 ['Yes' 'No']
 ['No' 'No']
 ...
 ['Yes' 'No']
 ['No' 'No']
 ['No' 'No']]
```

```
In [88]: print(dataframe)
```

```
      Rain on Tomorrow Prediction of Rain
0                  Yes                 No
1                  Yes                 No
2                   No                 No
3                   No                Yes
4                   No                 No
...                ...                ...
29087               No                Yes
29088               No                 No
29089              Yes                 No
29090               No                 No
29091               No                 No

[29092 rows x 2 columns]
```

Calculating Accuracy

```
In [89]: from sklearn.metrics import accuracy_score
         accuracy_score(Y_test,y_pred)
```

```
Out[89]: 0.8538086071772308
```

```
In [90]: dataframe.to_csv('prediction.csv')
```

```
In [ ]:
```