WEATHER PREDICTION

Predicting it will Rain or not using some Weather Conditons..

About Dataset

Using the Columns :

* precipitation
* temp_max
* temp_min
* wind

We are going to predict the weather condition :

* drizzle
* rain
* sun
* snow
* fog

seattle-weather.csv(49.68 kB)

get_app

fullscreen

chevron_right

Detail

Compact

Column

6 of 6 columns

keyboard_arrow_down

About this file

Dataset containing a Weather conditions Based on samples.

Sun

```
# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will l
ist all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that
gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be sav
ed outside of the current session
/kaggle/input/weather-prediction/seattle-weather.csv
```

DataSet:

Based on some factor we are, going to predict the weathers..

- date : dates
- precipitation : All forms in which water falls on the land surface and open water bodies as rain, sleet, snow, hail, or drizzle
- temp_max : Maximum Temperature
- temp_min : Minimum Temperature
- wind : Wind speed
- weather : weathers types

DataSet Link

https://www.kaggle.com/ananthr1/weather-prediction

In [2]:

```
#Import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
#Load the dataset
data = pd.read_csv("../input/weather-prediction/seattle-weather.csv")
```

In [4]:

linkcode

```
data.head()
```

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle

date	precipitation	temp_max	temp_min	wind	weather	
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain

```
data.tail()
```

Out[5]:

	date	precipitation	temp_max	temp_min	wind	weather
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 6 columns):
```

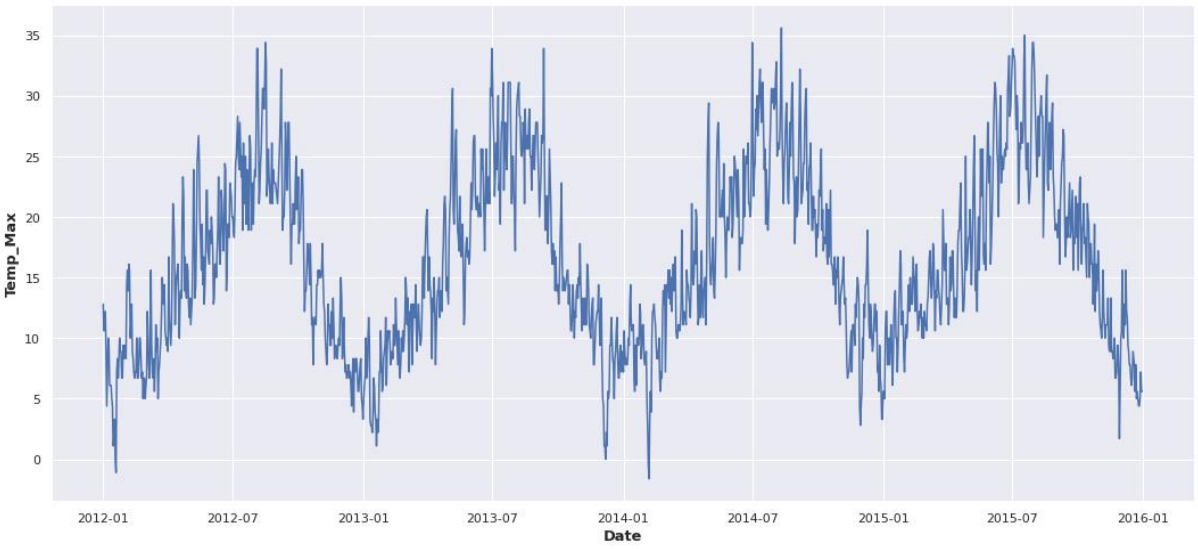
```
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           1461 non-null   object
 1   precipitation  1461 non-null   float64
 2   temp_max       1461 non-null   float64
 3   temp_min       1461 non-null   float64
 4   wind           1461 non-null   float64
 5   weather        1461 non-null   object
dtypes: float64(4), object(2)
memory usage: 68.6+ KB
```

In [7]:

```
#Check for null values
data.isnull().sum()
```

Out[7]:

```
date             0
precipitation    0
temp_max         0
temp_min         0
wind             0
weather          0
dtype: int64
```




```
#convert the data type into datetime
data['date'] = pd.to_datetime(data['date'])
```

In [9]:

```
data.nunique()
```

Out[9]:

```
date             1461
precipitation     111
temp_max           67
temp_min           55
wind               79
```

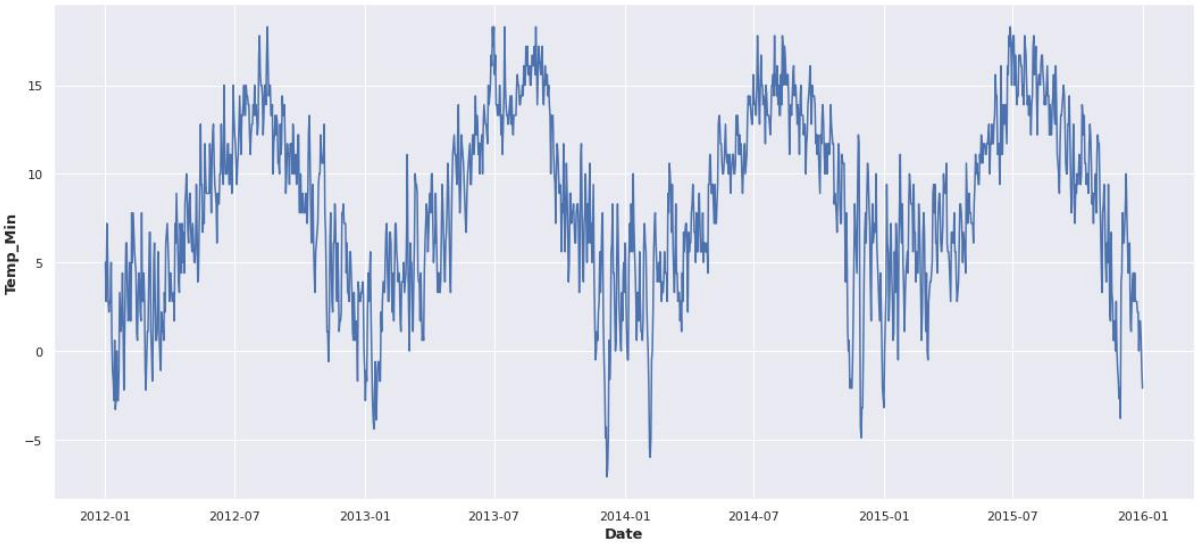
```
weather             5
dtype: int64
```

In [10]:

linkcode

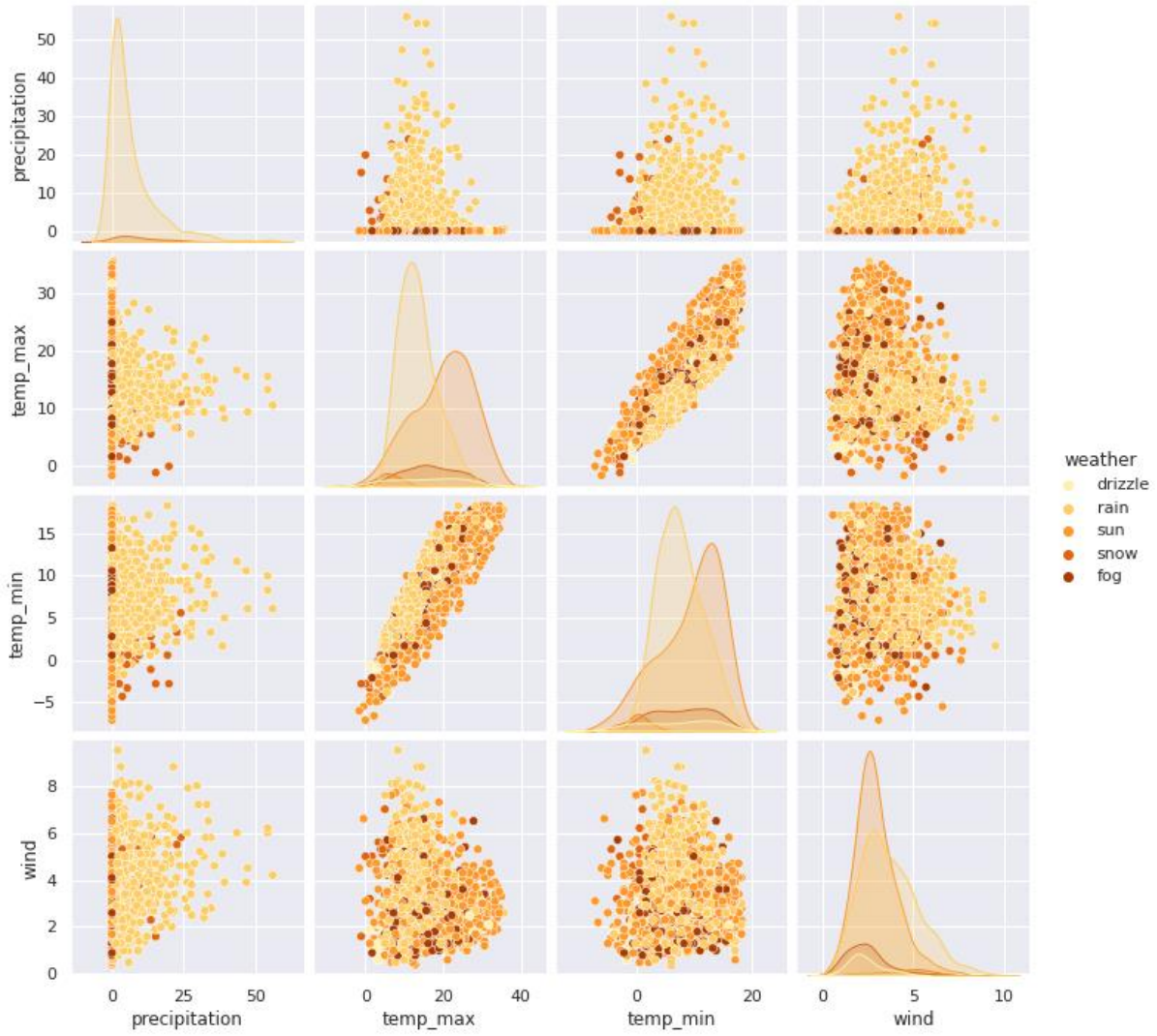
```
plt.figure(figsize=(10,5))
sns.set_theme()
sns.countplot(x = 'weather',data = data,palette="ch:start=.2,rot=-.3")
plt.xlabel("weather",fontweight='bold',size=13)
plt.ylabel("Count",fontweight='bold',size=13)
plt.show()
```

```
plt.figure(figsize=(18,8))
sns.set_theme()
sns.lineplot(x = 'date',y='temp_min',data=data)
plt.xlabel("Date",fontweight='bold',size=13)
plt.ylabel("Temp_Min",fontweight='bold',size=13)
plt.show()
```

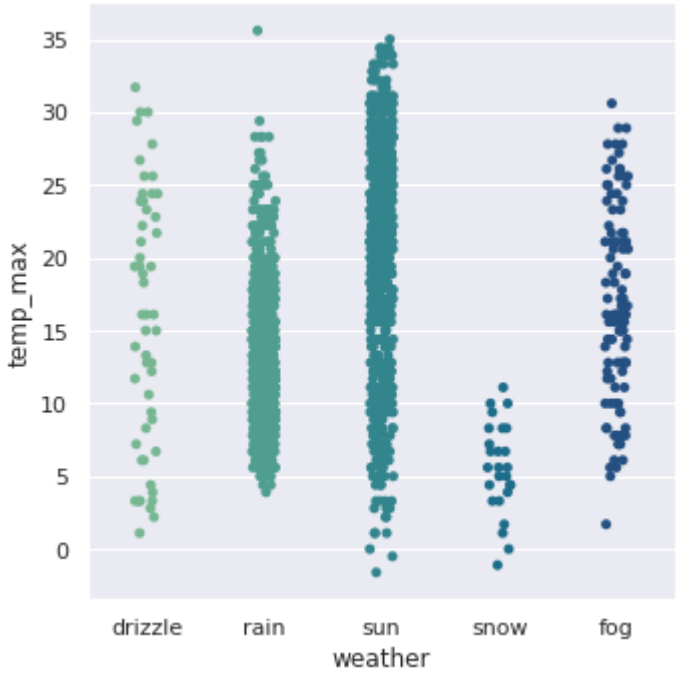



```
plt.figure(figsize=(14,8))
sns.pairplot(data.drop('date',axis=1),hue='weather',palette="YlOrBr")
plt.show()
```

```
<Figure size 1008x576 with 0 Axes>
```



<Figure size 720x360 with 0 Axes>

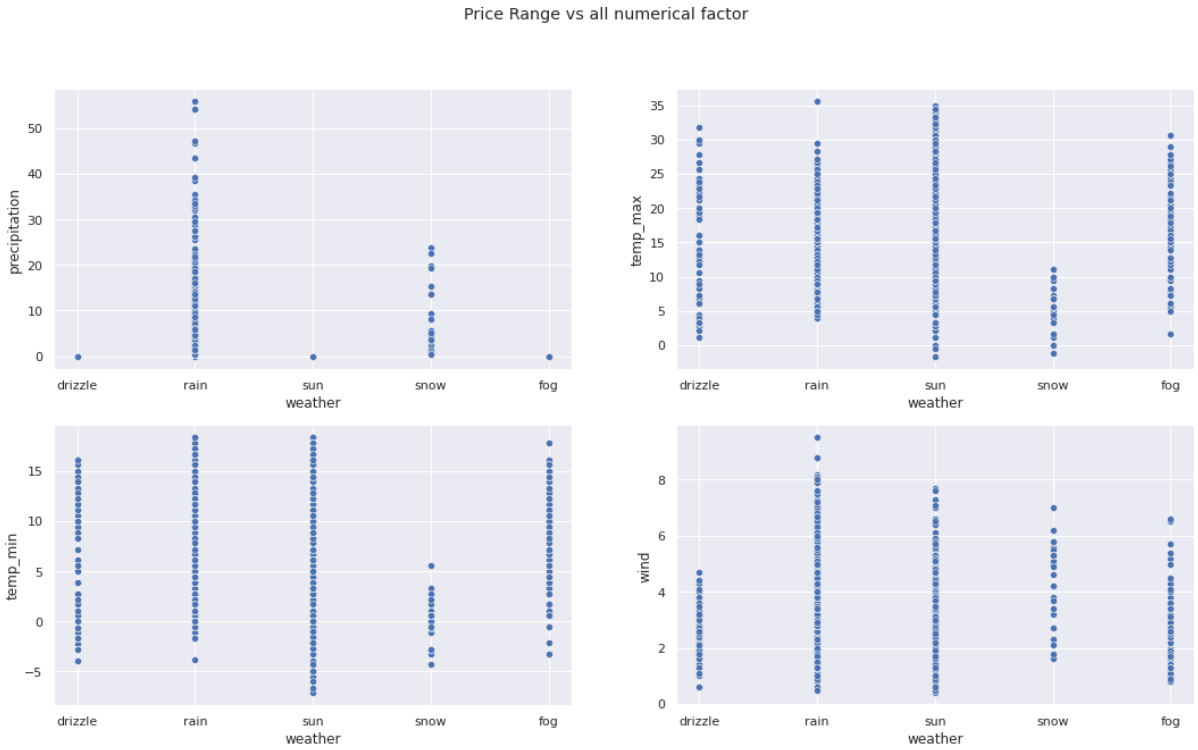



```
fig, axes = plt.subplots(2, 2, figsize=(18, 10))

fig.suptitle('Price Range vs all numerical factor')

sns.scatterplot(ax=axes[0, 0], data=data, x='weather', y='precipitation
')
sns.scatterplot(ax=axes[0, 1], data=data, x='weather', y='temp_max')
sns.scatterplot(ax=axes[1, 0], data=data, x='weather', y='temp_min')
sns.scatterplot(ax=axes[1, 1], data=data, x='weather', y='wind')
plt.show()
```

```
s.scatterplot(ax=axes[1, 0], data=data, x='weather', y='temp_min')
sns.scatterplot(ax=axes[1, 1], data=data, x='weather', y='wind')
plt.show()
```



In [20]:

```
def LABEL_ENCODING(c1):
```

```
def LABEL_ENCODING(c1):
    from sklearn import preprocessing
    label_encoder = preprocessing.LabelEncoder()
    data[c1]= label_encoder.fit_transform(data[c1])
    data[c1].unique()
LABEL_ENCODING("weather")
data
```

Out[20]:

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	0
1	2012-01-02	10.9	10.6	2.8	4.5	2
2	2012-01-03	0.8	11.7	7.2	2.3	2
3	2012-01-04	20.3	12.2	5.6	4.7	2

	date	precipitation	temp_max	temp_min	wind	weather
4	2012-01-05	1.3	8.9	2.8	6.1	2
...	...	...	...	...	...	...
1456	2015-12-27	8.6	4.4	1.7	2.9	2
1457	2015-12-28	1.5	5.0	1.7	1.3	2
1458	2015-12-29	0.0	7.2	0.6	2.6	1
1459	2015-12-30	0.0	5.6	-1.0	3.4	4
1460	2015-12-31	0.0	5.6	-2.1	3.5	4

1461 rows × 6 columns

In [21]:

```
data = data.drop('date',axis=1)
```

In [22]:

```
linkcode
x = data.drop('weather',axis=1)
y = data['weather']
```

Split the dataset into train and test

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)
```

In [24]:

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1095, 4)
```

```
(366, 4)
(1095,)
(366,)
```

Feature Scaling

In [25]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Logistic Regression

In [26]:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

Out[26]:

```
LogisticRegression(random_state=0)
```

In [27]:

```
y_pred = classifier.predict(X_test)
```

In [28]:

```
y_pred
```

Out[28]:

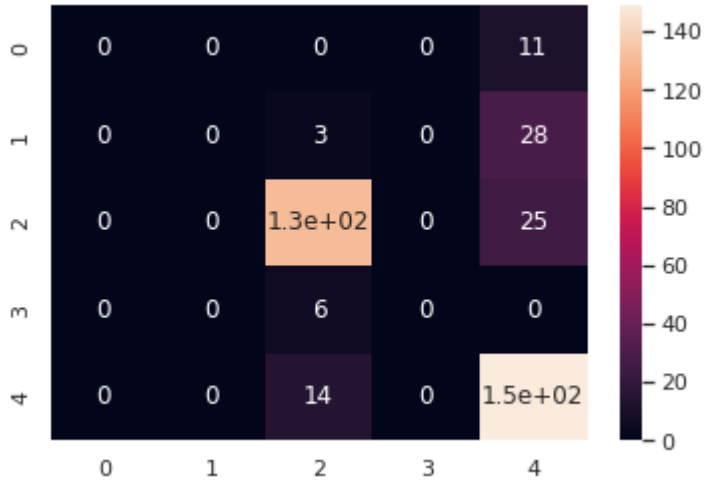
```
array([4, 2, 2, 4, 4, 2, 2, 2, 4, 4, 4, 2, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2
,
       2, 4, 4, 4, 4, 2, 4, 4, 2, 4, 2, 2, 2, 2, 4, 2, 4, 2, 4, 2, 2, 2
,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 2, 4, 2, 4, 4, 4, 4, 4, 2
,
       4, 4, 4, 2, 2, 2, 2, 4, 4, 4, 4, 4, 2, 4, 4, 4, 2, 2, 2, 4, 4, 2
,
       4, 4, 4, 4, 2, 4, 2, 2, 4, 4, 4, 4, 2, 2, 4, 2, 2, 4, 2, 2, 4, 4
,
       2, 4, 2, 4, 4, 4, 4, 2, 2, 2, 4, 2, 4, 4, 4, 4, 4, 4, 4, 2, 4, 2
,
       4, 2, 2, 2, 4, 2, 4, 4, 2, 2, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 2
,
       4, 2, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 4, 2, 2, 2, 4, 4, 4, 4, 2, 2
,
       4, 4, 4, 4, 4, 4, 2, 4, 2, 4, 4, 2, 2, 4, 4, 4, 4, 4, 2, 2, 4, 2
,
       4, 4, 2, 2, 4, 4, 2, 4, 2, 4, 4, 2, 2, 2, 2, 2, 4, 4, 4, 2, 2, 2
,
       4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 4, 2, 4, 2, 2, 4, 2
,
       4, 2, 2, 4, 4, 4, 4, 4, 4, 4, 2, 2, 4, 4, 4, 4, 4, 4, 2, 2, 4, 4
,
```

```
       4, 2, 4, 2, 4, 2, 4, 4, 2, 2, 4, 4, 4, 4, 2, 4, 2, 2, 2, 4, 4, 4
,
       4, 4, 2, 4, 4, 2, 4, 2, 4, 2, 2, 4, 2, 4, 2, 4, 4, 4, 2, 4, 4, 2
,
       2, 4, 2, 4, 2, 2, 4, 4, 2, 2, 2, 4, 2, 4, 2, 2, 4, 4, 2, 2, 2, 4
,
       4, 2, 4, 4, 2, 2, 4, 4, 2, 4, 2, 4, 4, 2, 4, 2, 2, 2, 4, 2, 4, 4
,
       4, 4, 4, 4, 2, 2, 4, 4, 4, 2, 4, 2, 2, 4])
```

In [29]:

```
linkcode
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
sns.heatmap(cm,annot=True)
plt.show()
```



Predict the tset set result

In [33]:

```
y_pred = classifier.predict(X_test)
```

In [34]:

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
acc2 = accuracy_score(y_test, y_pred)
```

```
[[  0   0   0   0  11]
 [  0   0   0   0  31]
 [  0   0 126   0  29]
 [  0   0   4   2   0]
 [  0   0   0   0 163]]
```

In [35]:

```
print(f"Accuracy score: {acc2}")
```

```
Accuracy score: 0.7950819672131147
```

Training the K-NN model on the Training set

In [36]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p
= 2)
classifier.fit(X_train, y_train)
```

Out[36]:

```
KNeighborsClassifier()
```

In [37]:

```
y_pred = classifier.predict(X_test)
```

In [38]:

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[  1   1   3   0   6]
 [  1   4   5   0  21]
 [  0   3 127   0  25]
 [  0   0   3   1   2]
 [  5  17  26   0 115]]
```

In [39]:

```
acc3 = accuracy_score(y_test, y_pred)
print(f"Accuracy score: {acc3}")
```

```
Accuracy score: 0.6775956284153005
```