```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import accuracy_score,ConfusionMatrixDisplay,classification_report
```

To read dataset

```
df=pd.read_csv('/content/drive/MyDrive/dataset/seattle-weather.csv')
df
```

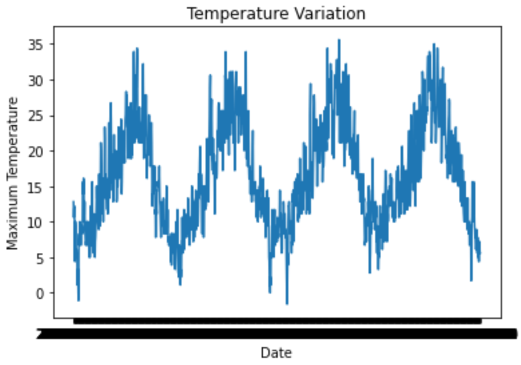
	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain
...	...	...	...	...	...	...
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	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

1461 rows × 6 columns

```
df.corr()
```

	precipitation	temp_max	temp_min	wind
precipitation	1.000000	-0.228555	-0.072684	0.328045
temp_max	-0.228555	1.000000	0.875687	-0.164857
temp_min	-0.072684	0.875687	1.000000	-0.074185
wind	0.328045	-0.164857	-0.074185	1.000000

```
x=df['date']
y=df['temp_max']
plt.plot(x,y)
plt.xlabel('Date')
plt.ylabel('Maximum Temperature')
plt.title('Temperature Variation')
plt.show()
```



To check datatypes:

```
df.dtypes
```

```
date             object
precipitation   float64
temp_max        float64
```

```
     temp_min         float64
     wind             float64
     weather           object
     dtype: object
```

To check null values:

```
df.isna().sum()
```

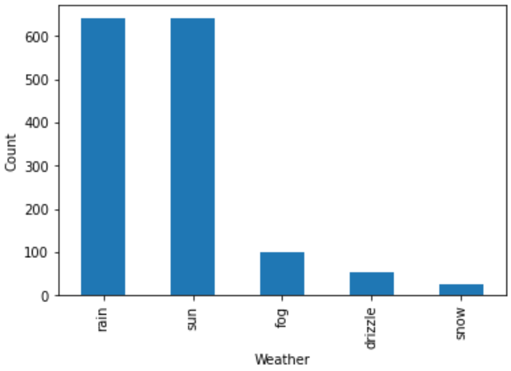
```
     date             0
     precipitation    0
     temp_max         0
     temp_min         0
     wind             0
     weather          0
     dtype: int64
```

```
c=df['weather'].value_counts()
c
```

```
     rain       641
     sun        640
     fog        101
     drizzle     53
     snow        26
     Name: weather, dtype: int64
```

```
y1=df['weather'].value_counts().plot(kind='bar')
y1.set_xlabel('Weather')
y1.set_ylabel('Count')
```

```
     Text(0, 0.5, 'Count')
```



To drop a column:

```
df.drop(['date'],axis=1,inplace=True)
```

To change datatypes:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['weather']=le.fit_transform(df['weather'])
```

```
df.dtypes
```

```
     precipitation    float64
     temp_max         float64
     temp_min         float64
     wind             float64
     weather            int64
     dtype: object
```

Seperating X and y:

```
X=df.iloc[:,:-1].values
X
```

```
     array([[ 0. , 12.8,  5. ,  4.7],
            [10.9, 10.6,  2.8,  4.5],
            [ 0.8, 11.7,  7.2,  2.3],
            ...,
            [ 0. ,  7.2,  0.6,  2.6],
            [ 0. ,  5.6, -1. ,  3.4],
            [ 0. ,  5.6, -2.1,  3.5]])
```

```
y=df.iloc[:,-1].values
y
```

```
    array([0, 2, 2, ..., 1, 4, 4])
```

```
df['weather'].value_counts()
```

```
    2    641
    4    640
    1    101
    0     53
    3     26
    Name: weather, dtype: int64
```

Scaling values:

```
from sklearn.preprocessing import MinMaxScaler
minmax=MinMaxScaler()
X_new=minmax.fit_transform(X)
X_new
```

```
    array([[0.        , 0.38709677, 0.47637795, 0.47252747],
           [0.19499106, 0.32795699, 0.38976378, 0.45054945],
           [0.01431127, 0.35752688, 0.56299213, 0.20879121],
           ...,
           [0.        , 0.23655914, 0.30314961, 0.24175824],
           [0.        , 0.19354839, 0.24015748, 0.32967033],
           [0.        , 0.19354839, 0.19685039, 0.34065934]])
```

Training and Testing Data:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_new,y,test_size=0.25,random_state=10)
```

```
X_train.shape
```

```
    (1095, 4)
```

```
y_train.shape
```

```
    (1095,)
```

Using KNN Algorithm:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)
ypred=knn.predict(X_test)
ypred
```

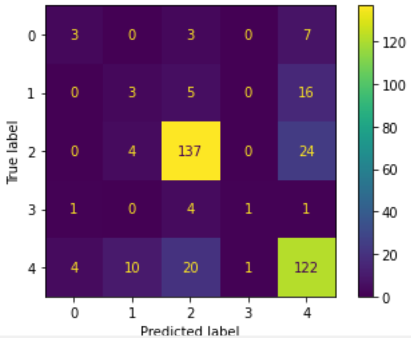
```
    array([2, 4, 2, 2, 2, 2, 4, 4, 4, 4, 4, 2, 4, 2, 2, 4, 4, 2, 4, 4, 4, 2,
           4, 2, 4, 2, 2, 4, 2, 1, 2, 4, 4, 4, 2, 2, 2, 4, 2, 4, 2, 4, 2, 4,
           0, 4, 4, 4, 2, 2, 4, 4, 2, 4, 4, 4, 2, 4, 4, 4, 1, 2, 2, 1, 4, 2,
           4, 2, 4, 4, 2, 2, 2, 4, 2, 4, 4, 4, 4, 2, 2, 4, 4, 2, 4, 4, 2, 2,
           4, 4, 3, 2, 2, 1, 4, 2, 4, 4, 4, 4, 4, 4, 2, 2, 2, 1, 4, 2, 4, 2,
           4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 2, 2, 2, 1, 4, 2, 4, 2, 4, 4,
           2, 4, 4, 4, 2, 2, 2, 2, 2, 4, 2, 2, 4, 2, 4, 4, 4, 2, 2, 4, 2, 4,
           4, 2, 4, 2, 4, 0, 4, 4, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 4, 4, 2, 4,
           4, 4, 2, 4, 4, 4, 4, 4, 2, 2, 4, 4, 2, 4, 4, 1, 2, 4, 2, 2, 2, 2,
           2, 2, 2, 4, 4, 4, 2, 1, 4, 0, 4, 4, 3, 0, 4, 2, 4, 4, 2, 2, 4, 4,
           4, 2, 4, 2, 2, 4, 4, 4, 2, 2, 2, 2, 2, 4, 4, 1, 4, 4, 2, 4, 4, 4,
           2, 4, 4, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 2, 2, 4, 2, 2, 2, 2, 2,
           4, 1, 4, 4, 2, 4, 2, 2, 4, 2, 4, 4, 4, 4, 4, 2, 2, 2, 4, 2, 2, 2,
           4, 2, 2, 2, 2, 0, 2, 2, 2, 2, 4, 2, 4, 2, 0, 1, 2, 4, 4, 1, 2, 2,
           4, 0, 4, 4, 4, 2, 2, 4, 4, 2, 1, 4, 2, 2, 1, 2, 4, 4, 4, 2, 2, 2,
           1, 2, 4, 2, 2, 4, 2, 1, 2, 0, 4, 4, 2, 1, 2, 2, 2, 2, 2, 2, 2, 4,
           2, 4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 4])
```

```
print(accuracy_score(y_test,ypred)*100)
```

```
    72.6775956284153
```

```
print(ConfusionMatrixDisplay.from_predictions(y_test,ypred))
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fc3e8fb9
```



```
print(classification_report(y_test,ypred))
```

```
              precision    recall  f1-score   support

           0       0.38      0.23      0.29        13
           1       0.18      0.12      0.15        24
           2       0.81      0.83      0.82       165
           3       0.50      0.14      0.22         7
           4       0.72      0.78      0.75       157

    accuracy                           0.73       366
   macro avg       0.52      0.42      0.44       366
weighted avg       0.71      0.73      0.71       366
```

Using SVM Algorithm:

```
from sklearn.svm import SVC
sv=SVC(kernel='rbf')
sv.fit(X_train,y_train)
y_pred=sv.predict(X_test)
y_pred
```

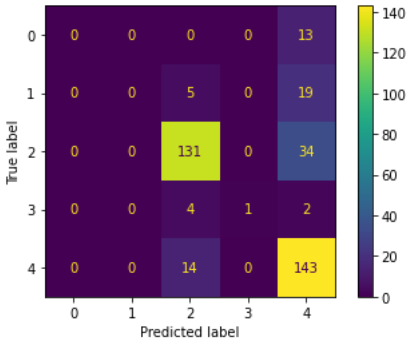
```
array([2, 4, 2, 4, 4, 2, 2, 4, 4, 4, 4, 2, 4, 2, 2, 4, 4, 2, 4, 4, 4, 2,
       4, 2, 4, 2, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 4, 4, 2, 4, 2, 4,
       4, 4, 4, 4, 2, 2, 4, 2, 2, 4, 4, 4, 2, 4, 4, 2, 4, 2, 2, 2, 4, 2,
       4, 2, 4, 4, 2, 2, 2, 4, 2, 4, 4, 4, 4, 2, 4, 4, 4, 2, 4, 4, 2, 2,
       4, 4, 3, 2, 2, 4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 2,
       4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 4, 4, 2, 4, 2, 4, 4,
       2, 4, 4, 2, 2, 2, 4, 2, 4, 4, 2, 2, 4, 4, 4, 4, 4, 2, 2, 4, 2, 4,
       4, 2, 4, 2, 4, 4, 4, 4, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 4, 4, 2, 4,
       4, 4, 2, 4, 4, 4, 4, 4, 2, 2, 4, 4, 2, 4, 4, 4, 2, 4, 2, 2, 2, 2,
       2, 4, 2, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 4, 2, 2, 4, 4,
       4, 2, 4, 2, 2, 4, 4, 4, 2, 2, 4, 2, 2, 4, 4, 4, 4, 4, 2, 4, 4, 4,
       4, 2, 4, 2, 2, 2, 4, 4, 4, 4, 4, 4, 2, 2, 4, 2, 4, 2, 4, 4, 2, 2,
       4, 4, 4, 4, 2, 2, 2, 4, 4, 2, 4, 4, 4, 4, 4, 2, 4, 2, 4, 2, 4, 2,
       4, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4, 2, 4, 4, 4, 2, 2, 4, 4, 4, 2, 2,
       4, 4, 4, 4, 4, 2, 2, 4, 4, 4, 4, 4, 2, 2, 2, 2, 4, 4, 4, 2, 2, 2,
       4, 2, 4, 2, 2, 4, 2, 4, 2, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 4,
       4, 4, 2, 4, 4, 2, 2, 4, 2, 2, 2, 2, 2, 4])
```

```
print(accuracy_score(y_test,y_pred)*100)
```

```
75.13661202185791
```

```
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fc3e968e040>
```



```
print(classification_report(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.00      0.00      0.00        13
               1       0.00      0.00      0.00        24
               2       0.85      0.79      0.82       165
               3       1.00      0.14      0.25         7
               4       0.68      0.91      0.78       157

        accuracy                           0.75       366
       macro avg       0.51      0.37      0.37       366
    weighted avg       0.69      0.75      0.71       366
```

Naive-Bayes Algorithm

```
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb.fit(X_train,y_train)
ypre=nb.predict(X_test)
ypre
```

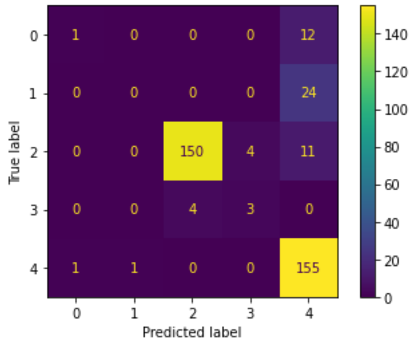
```
    array([2, 4, 2, 4, 4, 2, 4, 4, 4, 4, 4, 2, 4, 2, 2, 4, 4, 2, 3, 4, 4, 2,
           4, 2, 1, 2, 2, 4, 2, 4, 4, 2, 2, 4, 2, 4, 2, 4, 4, 4, 2, 4, 2, 4,
           4, 4, 4, 4, 2, 2, 2, 4, 2, 2, 4, 4, 2, 4, 4, 2, 4, 2, 2, 2, 4, 2,
           4, 2, 4, 4, 2, 2, 2, 4, 4, 4, 4, 4, 2, 2, 4, 4, 4, 2, 4, 4, 2, 2,
           4, 4, 3, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 4, 2, 4, 2,
           4, 2, 4, 4, 2, 4, 4, 4, 4, 4, 2, 4, 2, 2, 2, 4, 4, 2, 4, 2, 4, 4,
           2, 2, 4, 2, 2, 2, 4, 2, 4, 4, 2, 2, 4, 2, 4, 4, 4, 2, 2, 4, 2, 4,
           4, 2, 4, 2, 4, 4, 4, 4, 2, 2, 2, 2, 4, 4, 2, 2, 2, 2, 4, 4, 2, 4,
           2, 2, 2, 4, 4, 4, 4, 2, 2, 2, 4, 4, 2, 4, 4, 4, 2, 4, 2, 2, 2, 2,
           3, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 3, 4, 2, 4, 4, 2, 2, 4, 4,
           4, 4, 2, 2, 2, 4, 4, 4, 2, 2, 4, 4, 2, 4, 4, 4, 4, 4, 3, 4, 4, 4,
           2, 2, 4, 4, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 2, 2, 2,
           4, 4, 4, 4, 2, 4, 2, 3, 4, 2, 4, 4, 4, 4, 0, 2, 4, 2, 2, 2, 4, 2,
           4, 2, 2, 2, 2, 4, 2, 2, 2, 4, 4, 2, 4, 4, 4, 2, 2, 4, 4, 4, 2, 2,
           4, 0, 4, 2, 4, 2, 2, 4, 4, 4, 4, 4, 2, 2, 2, 2, 4, 4, 4, 2, 2, 2,
           4, 4, 4, 2, 2, 4, 4, 4, 2, 4, 4, 4, 2, 4, 4, 2, 2, 2, 2, 2, 3, 2,
           4, 4, 4, 4, 4, 2, 2, 2, 2, 4, 2, 2, 2, 4])
```

```
print(accuracy_score(y_test,ypre)*100)
```

```
    84.42622950819673
```

```
print(ConfusionMatrixDisplay.from_predictions(y_test,ypre))
```

```
    <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fc3ec4f1e20>
```



```
print(classification_report(y_test,ypre))
```

```
                  precision    recall  f1-score   support

               0       0.50      0.08      0.13        13
               1       0.00      0.00      0.00        24
               2       0.97      0.91      0.94       165
               3       0.43      0.43      0.43         7
               4       0.77      0.99      0.86       157

        accuracy                           0.84       366
       macro avg       0.53      0.48      0.47       366
    weighted avg       0.79      0.84      0.81       366
```

Oversampling Technique

```
from imblearn.over_sampling import SMOTE
oversample=SMOTE(random_state=1)
```

```
X_os,y_os=oversample.fit_resample(X_new,y)
```

```
np.unique(y_os, return_counts=True)
```

```
    (array([0, 1, 2, 3, 4]), array([641, 641, 641, 641, 641]))
```

```
from collections import Counter
print(Counter(y_os))
```

```
    Counter({0: 641, 2: 641, 4: 641, 3: 641, 1: 641})
```

```
X_trainos,X_testos,y_trainos,y_testos=train_test_split(X_os,y_os,test_size=0.2,random_state=1)
```

```
knnos=KNeighborsClassifier(n_neighbors=5)
knnos.fit(X_trainos,y_trainos)
y_predos=knnos.predict(X_testos)
y_predos
```

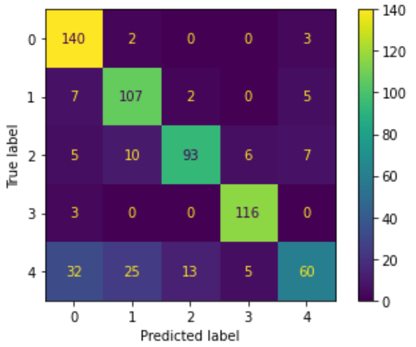
```
    array([0, 3, 1, 4, 1, 2, 0, 0, 2, 3, 2, 4, 3, 2, 2, 1, 3, 1, 2, 4, 4, 0,
           1, 0, 2, 3, 2, 0, 3, 4, 1, 1, 4, 0, 3, 4, 3, 3, 3, 1, 2, 1, 0, 2,
           0, 1, 0, 3, 4, 3, 0, 0, 1, 4, 3, 0, 0, 1, 0, 0, 1, 2, 2, 0, 1, 1,
           0, 2, 2, 2, 4, 4, 1, 4, 0, 3, 0, 0, 0, 1, 3, 3, 0, 1, 0, 2, 3, 3,
           0, 0, 0, 1, 0, 4, 4, 4, 1, 0, 1, 0, 0, 0, 2, 2, 0, 0, 1, 1, 1, 0,
           4, 0, 1, 3, 0, 2, 2, 3, 3, 3, 2, 2, 0, 4, 2, 1, 0, 3, 1, 0, 1, 0,
           0, 3, 1, 1, 1, 1, 2, 0, 0, 2, 0, 2, 2, 1, 3, 3, 2, 1, 3, 2, 1, 1,
           0, 2, 1, 4, 4, 0, 4, 3, 3, 1, 4, 1, 3, 1, 3, 2, 0, 3, 0, 0, 3, 4,
           0, 1, 2, 2, 2, 4, 1, 4, 0, 3, 1, 3, 0, 4, 3, 1, 3, 0, 2, 1, 2, 4,
           2, 0, 3, 4, 2, 2, 0, 3, 1, 1, 0, 1, 4, 1, 3, 2, 1, 2, 0, 1, 3, 2,
           0, 3, 3, 3, 3, 0, 3, 0, 2, 2, 0, 4, 3, 1, 0, 2, 4, 3, 0, 0, 1, 2,
           3, 3, 2, 0, 1, 0, 1, 1, 1, 3, 0, 0, 0, 4, 1, 2, 3, 3, 2, 2, 0, 1,
           0, 0, 1, 4, 4, 4, 0, 0, 2, 3, 0, 1, 3, 3, 0, 3, 4, 2, 2, 1, 3, 0,
           1, 0, 4, 4, 0, 1, 1, 1, 2, 2, 3, 0, 0, 3, 0, 0, 0, 0, 3, 4, 0, 3,
           4, 4, 3, 0, 0, 0, 2, 1, 2, 3, 1, 1, 2, 3, 1, 1, 1, 3, 3, 1, 3, 1,
           2, 3, 3, 3, 2, 3, 0, 0, 0, 4, 0, 1, 1, 0, 2, 0, 4, 0, 0, 4, 3, 3,
           1, 0, 3, 1, 2, 3, 0, 2, 4, 2, 4, 1, 2, 0, 0, 3, 2, 1, 0, 4, 0, 1,
           0, 0, 1, 2, 1, 3, 0, 3, 2, 2, 1, 2, 0, 0, 1, 0, 3, 0, 0, 2, 4, 0,
           2, 2, 2, 2, 0, 0, 0, 2, 2, 1, 3, 3, 3, 0, 0, 4, 0, 4, 0, 0, 1, 4,
           3, 4, 0, 3, 1, 0, 0, 1, 1, 0, 3, 3, 0, 3, 1, 0, 4, 0, 0, 1, 0, 4,
           4, 0, 2, 1, 4, 2, 0, 1, 3, 2, 3, 1, 0, 0, 3, 0, 4, 0, 0, 0, 3, 3,
           0, 1, 3, 4, 4, 0, 2, 1, 1, 3, 2, 2, 2, 4, 1, 0, 3, 3, 1, 1, 1, 1,
           0, 3, 0, 0, 1, 0, 3, 1, 3, 4, 0, 0, 0, 3, 0, 3, 0, 2, 1, 1, 0, 0,
           0, 0, 2, 1, 2, 2, 1, 0, 1, 0, 1, 1, 0, 1, 4, 1, 1, 3, 2, 1, 1, 0,
           3, 3, 0, 2, 3, 2, 4, 4, 2, 0, 1, 0, 0, 4, 2, 0, 2, 4, 1, 0, 0, 4,
           1, 1, 0, 4, 1, 0, 4, 2, 1, 4, 1, 3, 0, 3, 2, 1, 2, 3, 0, 2, 4, 3,
           0, 3, 1, 3, 1, 3, 3, 0, 0, 0, 2, 2, 0, 0, 4, 3, 2, 0, 3, 1, 1, 1,
           1, 3, 4, 1, 2, 4, 4, 1, 3, 1, 3, 0, 1, 3, 3, 2, 3, 2, 0, 0, 1, 0,
           3, 1, 1, 0, 1, 3, 1, 2, 1, 2, 4, 1, 0, 1, 0, 0, 0, 2, 1, 2, 0, 3,
           0, 3, 3])
```

```
print(accuracy_score(y_testos,y_predos)*100)
```

```
    80.49921996879876
```

```
print(ConfusionMatrixDisplay.from_predictions(y_testos,y_predos))
```

```
    <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fc3e843f040>
```



```
print(classification_report(y_testos,y_predos))
```

```
                  precision    recall  f1-score   support

               0       0.75      0.97      0.84       145
               1       0.74      0.88      0.81       121
               2       0.86      0.77      0.81       121
               3       0.91      0.97      0.94       119
               4       0.80      0.44      0.57       135

        accuracy                           0.80       641
```

```
       macro avg       0.81      0.81      0.80       641
    weighted avg       0.81      0.80      0.79       641
```

Undersampling Technique

```
from imblearn.under_sampling import RandomUnderSampler
undersample=RandomUnderSampler(random_state=40)
X_us,y_us=undersample.fit_resample(X_new,y)
```

```
np.unique(y_us, return_counts=True)
```

```
    (array([0, 1, 2, 3, 4]), array([26, 26, 26, 26, 26]))
```

```
X_trainus,X_testus,y_trainus,y_testus=train_test_split(X_us,y_us,random_state=20,test_size=0.2)
```

```
knnus=KNeighborsClassifier(n_neighbors=5)
knnus.fit(X_trainus,y_trainus)
y_predus=knnus.predict(X_testus)
y_predus
```

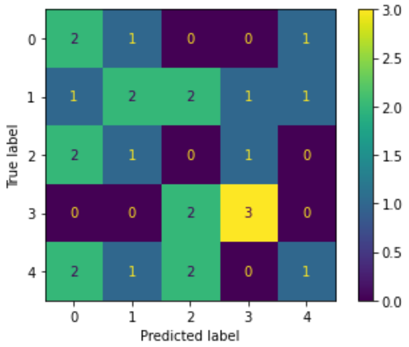
```
    array([0, 4, 1, 4, 3, 4, 0, 1, 2, 3, 0, 3, 0, 3, 0, 1, 1, 2, 2, 3, 2, 2,
           1, 0, 0, 2])
```

```
print(accuracy_score(y_testus,y_predus)*100)
```

```
    30.76923076923077
```

```
print(ConfusionMatrixDisplay.from_predictions(y_testus,y_predus))
```

```
    <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fc3e83ae280>
```



```
print(classification_report(y_testus,y_predus))
```

```
                  precision    recall  f1-score   support

               0       0.29      0.50      0.36         4
               1       0.40      0.29      0.33         7
               2       0.00      0.00      0.00         4
               3       0.60      0.60      0.60         5
               4       0.33      0.17      0.22         6

        accuracy                           0.31        26
       macro avg       0.32      0.31      0.30        26
    weighted avg       0.34      0.31      0.31        26
```

✓ 0s completed at 4:28 PM ● ×