```
import pandas as pd

import numpy as np


from google.colab import drive

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df = pd.read_csv('/content/drive/MyDrive/Prifina/FitBit_data.csv')

df
```

	ActivityDate	TotalSteps	TotalDistance	Active Status
0	3/25/2016	11004	7.110000	Highly Active
1	3/26/2016	17609	11.550000	Highly Active
2	3/27/2016	12736	8.530000	Highly Active
3	3/28/2016	13231	8.930000	Highly Active
4	3/29/2016	12041	7.850000	Highly Active
...	...	...	...	...
452	04-08-2016	23014	20.389999	Highly Active
453	04-09-2016	16470	8.070000	Highly Active
454	04-10-2016	28497	27.530001	Highly Active
455	04-11-2016	10622	8.060000	Highly Active
456	04-12-2016	2350	1.780000	low active

457 rows × 4 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 457 entries, 0 to 456
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   ActivityDate   457 non-null    object 
 1   TotalSteps     457 non-null    int64  
 2   TotalDistance  457 non-null    float64
 3   Active Status  457 non-null    object 
dtypes: float64(1), int64(1), object(2)
memory usage: 14.4+ KB
```

```
df.shape
```

```
(457, 4)
```

```
df.head()
```

	ActivityDate	TotalSteps	TotalDistance	Active Status
0	3/25/2016	11004	7.11	Highly Active
1	3/26/2016	17609	11.55	Highly Active
2	3/27/2016	12736	8.53	Highly Active
3	3/28/2016	13231	8.93	Highly Active
4	3/29/2016	12041	7.85	Highly Active

```
df.tail()
```

	ActivityDate	TotalSteps	TotalDistance	Active Status
452	04-08-2016	23014	20.389999	Highly Active
453	04-09-2016	16470	8.070000	Highly Active
454	04-10-2016	28497	27.530001	Highly Active
455	04-11-2016	10622	8.060000	Highly Active
456	04-12-2016	2350	1.780000	low active

```
df.isnull().sum()
```

```
ActivityDate     0
TotalSteps       0
TotalDistance    0
Active Status    0
dtype: int64
```

```
#replacing values in Active Status
```

```
df['Active Status'].replace(['low active','Active','Highly Active'],[0,1,2], inp
```

```
df.head()
```

	ActivityDate	TotalSteps	TotalDistance	Active Status
0	3/25/2016	11004	7.11	2
1	3/26/2016	17609	11.55	2
2	3/27/2016	12736	8.53	2
3	3/28/2016	13231	8.93	2
4	3/29/2016	12041	7.85	2

```
df.tail()
```

	ActivityDate	TotalSteps	TotalDistance	Active Status
452	04-08-2016	23014	20.389999	2
453	04-09-2016	16470	8.070000	2
454	04-10-2016	28497	27.530001	2
455	04-11-2016	10622	8.060000	2
456	04-12-2016	2350	1.780000	0

```
df['Active Status'].value_counts().sort_values(ascending=True)
```

```
1     42
2    126
0    289
Name: Active Status, dtype: int64
```

```
df.dtypes
```

```
ActivityDate      object
TotalSteps         int64
TotalDistance    float64
Active Status      int64
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder
```

```
lb_make = LabelEncoder()
```

```
df['ActivityDate'] = lb_make.fit_transform(df['ActivityDate'])
```

```
df
```

	ActivityDate	TotalSteps	TotalDistance	Active Status
0	25	11004	7.110000	2
1	26	17609	11.550000	2
2	27	12736	8.530000	2
3	28	13231	8.930000	2
4	29	12041	7.850000	2
...	...	...	...	...
452	8	23014	20.389999	2
453	9	16470	8.070000	2
454	10	28497	27.530001	2
455	11	10622	8.060000	2
456	12	2350	1.780000	0

457 rows × 4 columns

```
df.dtypes
```

```
ActivityDate       int64
TotalSteps         int64
TotalDistance    float64
Active Status      int64
dtype: object
```

```
new_df=df
```

```
x = new_df.drop(['Active Status'],axis=1)
```

```
x
```

	ActivityDate	TotalSteps	TotalDistance
0	25	11004	7.110000
1	26	17609	11.550000
2	27	12736	8.530000
3	28	13231	8.930000
4	29	12041	7.850000
...	...	...	...
452	8	23014	20.389999
453	9	16470	8.070000
454	10	28497	27.530001
455	11	10622	8.060000
456	12	2350	1.780000

457 rows × 3 columns

```
y = new_df['Active Status']
```

```
y
```

```
0      2
1      2
2      2
3      2
4      2
      ..
452    2
453    2
454    2
455    2
456    0
Name: Active Status, Length: 457, dtype: int64
```

```
from sklearn import preprocessing
```

```
scaler = preprocessing.StandardScaler()
```

```
df2 = pd.DataFrame(scaler.fit_transform(x))
```

```
df2
```

	0	1	2
0	1.985128	0.826587	0.599979
1	2.110467	2.051417	1.688854
2	2.235806	1.147769	0.948223
3	2.361145	1.239561	1.046320
4	2.486484	1.018888	0.781458
...	...	...	...
452	-0.145635	3.053720	3.856794
453	-0.020296	1.840201	0.835411
454	0.105043	4.070486	5.607823
455	0.230382	0.755749	0.832959
456	0.355721	-0.778210	-0.707161

457 rows × 3 columns

```
from sklearn.model_selection import train_test_split

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score

knn = KNeighborsClassifier(n_neighbors=10)

knn.fit(x_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

```
y_pred = knn.predict(x_test)

accuracy_score(y_test, y_pred)
```

```
1.0
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test,y_pred)
```

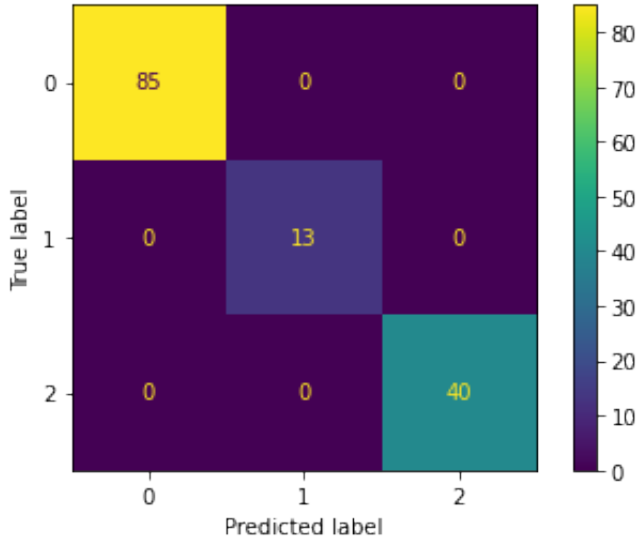
```
array([[85,  0,  0],
       [ 0, 13,  0],
       [ 0,  0, 40]])
```

```
from sklearn.metrics import ConfusionMatrixDisplay

disp = ConfusionMatrixDisplay(confusion_matrix(y_test,y_pred))
```

```
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f775ac754c0>
```




```
from sklearn.metrics import classification_report
```

```
from sklearn import metrics
```

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

```
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00        85
               1       1.00      1.00      1.00        13
               2       1.00      1.00      1.00        40

        accuracy                           1.00       138
       macro avg       1.00      1.00      1.00       138
    weighted avg       1.00      1.00      1.00       138
```

```
from sklearn.naive_bayes import GaussianNB
```

```
from sklearn.metrics import accuracy_score
```

```
gau = GaussianNB()
```

```
gau.fit(x_train,y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
y_pred = gau.predict(x_test)
```

```
accuracy_score(y_test,y_pred)
```

```
0.9347826086956522
```

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_test,y_pred)
```

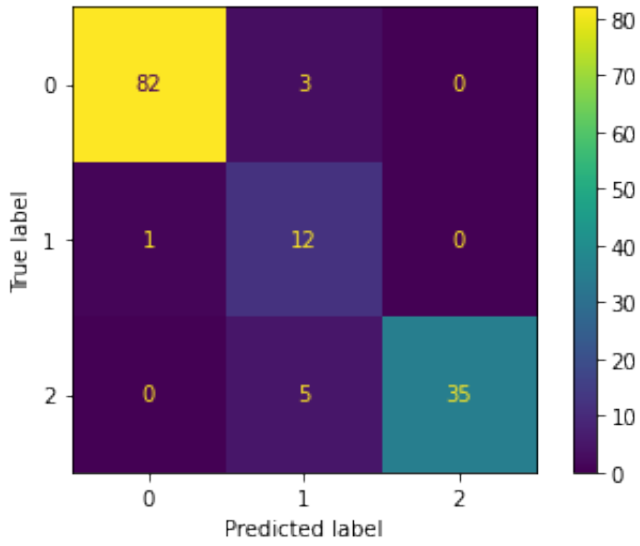
```
array([[82,  3,  0],
       [ 1, 12,  0],
       [ 0,  5, 35]])
```

```
from sklearn.metrics import ConfusionMatrixDisplay
```

```
disp = ConfusionMatrixDisplay(confusion_matrix(y_test,y_pred))
```

```
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f7758208bb0>
```



```
from sklearn.metrics import classification_report
```

```
from sklearn import metrics
```

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

```
                  precision    recall  f1-score   support

               0       0.99      0.96      0.98        85
               1       0.60      0.92      0.73        13
               2       1.00      0.88      0.93        40

        accuracy                           0.93       138
       macro avg       0.86      0.92      0.88       138
    weighted avg       0.95      0.93      0.94       138
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score
```

```
rf = RandomForestClassifier()
```

```
rf.fit(x_train,y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
y_pred = rf.predict(x_test)
```

```
accuracy_score(y_test,y_pred)
```

```
0.9927536231884058
```

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_test,y_pred)
```

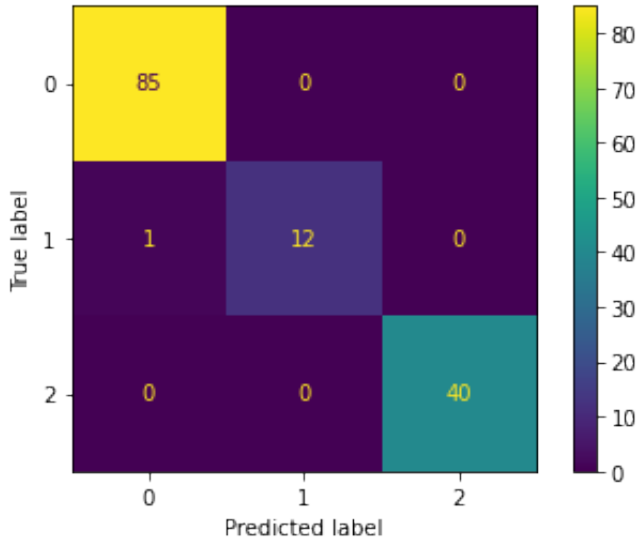
```
array([[85,  0,  0],
       [ 1, 12,  0],
       [ 0,  0, 40]])
```

```
from sklearn.metrics import ConfusionMatrixDisplay
```

```
disp = ConfusionMatrixDisplay(confusion_matrix(y_test,y_pred))
```

```
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f775ac756a0>
```




```
from sklearn.metrics import classification_report
```

```
from sklearn import metrics
```

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

```
                  precision    recall  f1-score   support

               0       0.99      1.00      0.99        85
               1       1.00      0.92      0.96        13
               2       1.00      1.00      1.00        40

        accuracy                           0.99       138
       macro avg       1.00      0.97      0.98       138
    weighted avg       0.99      0.99      0.99       138
```