In [1]:

```
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
# 1.Load the kinematics dataset as measured on mobile sensors from the file “r
un_or_walk.csv”. List out the columns in the dataset.
data = pd.read_csv("run_or_walk.csv")
data.head()
```

Out[2]:

	date	time	username	wrist	activity	acceleration_x	acceleration_y	acceleratio
0	30-06-2017	13:51:15:847724020	viktor	0	0	0.2650	-0.7814	-0.0
1	30-06-2017	13:51:16:246945023	viktor	0	0	0.6722	-1.1233	-0.2
2	30-06-2017	13:51:16:446233987	viktor	0	0	0.4399	-1.4817	0.0
3	30-06-2017	13:51:16:646117985	viktor	0	0	0.3031	-0.8125	0.0
4	30-06-2017	13:51:16:846738994	viktor	0	0	0.4814	-0.9312	0.0

In [3]:

```
# 2. Let the target variable ‘y’ be the activity and assign all the columns af
ter it to ‘x’.
X = data.iloc[:,5:]
Y = data["activity"]
```

In [5]:

```
# 3.Using Scikit-learn fit a Gaussian Naive Bayes model and observe the accura
cy.Generate a classification report using scikitlearn.
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, rando
m_state=10)

g_model = GaussianNB()
g_model.fit(x_train, y_train)

predicted_values = g_model.predict(x_test)
print("\nAccuracy_Score\n")
print(metrics.accuracy_score(predicted_values, y_test))
```

```
Accuracy_Score

0.9580840576438274
```

In [6]:

```
print("\nClassification Report\n")
print(metrics.classification_report(predicted_values, y_test))
```

```
Classification Report

              precision    recall  f1-score   support

           0       0.99      0.93      0.96     14115
           1       0.93      0.99      0.96     12462

    accuracy                           0.96     26577
   macro avg       0.96      0.96      0.96     26577
weighted avg       0.96      0.96      0.96     26577
```

In [8]:

```
# 4.Repeat the model once using only the acceleration values as predictors and
then using only the gyro values as predictors.
# Comment on the difference in accuracy between both the models.

# Acceleration as independent variable
X_A = data.iloc[:,5:8]
Y_A = data["activity"]

x_train, x_test, y_train, y_test = train_test_split(X_A, Y_A, test_size=0.3, r
andom_state=10)
g_model = GaussianNB()
g_model.fit(x_train, y_train)

predicted_values = g_model.predict(x_test)
print("\nAccuracy_Score\n")
print(metrics.accuracy_score(predicted_values, y_test))

print("\nClassification Report\n")
print(metrics.classification_report(predicted_values, y_test))
```

```
Accuracy_Score

0.958648455431388

Classification Report

              precision    recall  f1-score   support

           0       0.99      0.93      0.96     14158
           1       0.92      0.99      0.96     12419

    accuracy                           0.96     26577
   macro avg       0.96      0.96      0.96     26577
weighted avg       0.96      0.96      0.96     26577
```

In [10]:

```
# Gyro as independent variable
X_G = data.iloc[:,8:]
Y_G = data["activity"]

x_train, x_test, y_train, y_test = train_test_split(X_G, Y_G, test_size=0.3, r
andom_state=10)
g_model = GaussianNB()
g_model.fit(x_train, y_train)

predicted_values = g_model.predict(x_test)
print("\nAccuracy_Score\n")
print(metrics.accuracy_score(predicted_values, y_test))

print("\nClassification Report\n")
print(metrics.classification_report(predicted_values, y_test))
```

```
Accuracy_Score

0.6486811905030666

Classification Report

              precision    recall  f1-score   support

           0       0.74      0.62      0.68     15810
           1       0.55      0.69      0.61     10767

    accuracy                           0.65     26577
   macro avg       0.65      0.65      0.65     26577
weighted avg       0.67      0.65      0.65     26577
```

In [ ]:

```
```