Assignment: Classify Kinematic Data

The comments/sections provided are your cues to perform the assignment. You don't need to limit yourself to the number of rows/cells provided. You can add additional rows in each section to add more lines of code.

If at any point in time you need help on solving this assignment, view our demo video to understand the different steps of the code.

Happy coding!

Classify Kinematic Data

DESCRIPTION

You are supposed to detect whether the person is running or walking based on the sensor data collected from iOS device. The dataset contains a single file which represents sensor data samples collected from accelerometer and gyroscope from iPhone 5c in 10 seconds interval and ~5.4/second frequency.

Objective: Practice classification based on Naive Bayes algorithm. Identify the predictors that can be influential.

Actions to Perform:

1. Load the kinematics dataset as measured on mobile sensors from the file "run_or_walk.csv."
2. List the columns in the dataset.
3. Let the target variable "y" be the activity, and assign all the columns after it to "x."
4. Using Scikit-learn, fit a Gaussian Naive Bayes model and observe the accuracy.
5. Generate a classification report using Scikit-learn.
6. Repeat the model once using only the acceleration values as predictors and then using only the gyro values as predictors.
7. Comment on the difference in accuracy between both models.

Load the kinematics dataset as measured on mobile sensors from the file "run_or_walk.csv."

In [1]:
```
#import required libraries and dataset
import pandas as pd
import matplotlib.pyplot as plot
%matplotlib inline

df = pd.read_csv("run_or_walk.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88588 entries, 0 to 88587
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   date            88588 non-null  object 
 1   time            88588 non-null  object 
 2   username        88588 non-null  object 
 3   wrist           88588 non-null  int64  
 4   activity        88588 non-null  int64  
 5   acceleration_x  88588 non-null  float64
 6   acceleration_y  88588 non-null  float64
 7   acceleration_z  88588 non-null  float64
 8   gyro_x          88588 non-null  float64
 9   gyro_y          88588 non-null  float64
 10  gyro_z          88588 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 6.4+ MB
```

In [2]:
```
df.isnull()
```

Out[2]:

	date	time	username	wrist	activity	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	g
0	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	
88583	False	False	False	False	False	False	False	False	False	False	
88584	False	False	False	False	False	False	False	False	False	False	
88585	False	False	False	False	False	False	False	False	False	False	
88586	False	False	False	False	False	False	False	False	False	False	
88587	False	False	False	False	False	False	False	False	False	False	

88588 rows × 11 columns

List the columns in the dataset.

In [3]:
```
df.columns
```

Out[3]:
```
Index(['date', 'time', 'username', 'wrist', 'activity', 'acceleration_x',
       'acceleration_y', 'acceleration_z', 'gyro_x', 'gyro_y', 'gyro_z'],
      dtype='object')
```

Let the target variable "y" be the activity, and assign all the columns after it to "X"

In [4]:
```
#Split the dataset in to training and testing sets
from sklearn.model_selection import train_test_split
X, y = df.iloc[:, 5:].values,df.iloc[:, 4].values
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [6]:
```
print(X_train.shape)
print(y_test[0:10])
```

```
(70870, 6)
[1 0 0 1 1 1 0 1 1 1]
```

Using Scikit-learn, fit a Gaussian Naive Bayes model and observe the accuracy.

In [7]:
```
#Fit the training data into Gaussian Naive Bayes classifier
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train,y_train)
y_predict = classifier.predict(X_test)
```

In [8]:
```
#print the accuracy score
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_predict,y_test)
print(accuracy)
```

```
0.9554690145614629
```

In [9]:
```
#print the confusion matrix
from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(y_predict,y_test)
print(conf_mat)
```

```
[[8583  699]
 [  90 8346]]
```

Generate a classification report using Scikit-learn

In [10]:
```
#print the classification report
from sklearn.metrics import classification_report
target_names = ["Walk","Run"]
print(classification_report(y_test, y_predict, target_names=target_names))
```

```
              precision    recall  f1-score   support

        Walk       0.92      0.99      0.96      8673
         Run       0.99      0.92      0.95      9045

    accuracy                           0.96     17718
   macro avg       0.96      0.96      0.96     17718
weighted avg       0.96      0.96      0.96     17718
```

Repeat the model once using only the acceleration values as predictors and then using only the gyro values as predictors.

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88588 entries, 0 to 88587
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   date            88588 non-null  object 
 1   time            88588 non-null  object 
 2   username        88588 non-null  object 
 3   wrist           88588 non-null  int64  
 4   activity        88588 non-null  int64  
 5   acceleration_x  88588 non-null  float64
 6   acceleration_y  88588 non-null  float64
 7   acceleration_z  88588 non-null  float64
 8   gyro_x          88588 non-null  float64
 9   gyro_y          88588 non-null  float64
 10  gyro_z          88588 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 6.4+ MB
```

Acceleration values as predictors

In [12]:
```
#Repeat the model once using Acceleration values as predictors
from sklearn.model_selection import train_test_split
X, y = df.iloc[:, [5,6,7]].values,df.iloc[:, 4].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [13]:
```
classifier.fit(X_train,y_train)
y_predict = classifier.predict(X_test)
#print the accuracy
print("Accuracy:",accuracy_score(y_predict,y_test))
```

```
Accuracy: 0.9565978101365843
```

In [14]:
```
#print the confusion matrix
conf_mat = confusion_matrix(y_predict,y_test)
print(conf_mat)
```

```
[[8610  706]
 [  63 8339]]
```

Gyro values as predictors

In [15]:
```
#Repeat the model once using Gyro values as predictors
X, y = df.iloc[:, [8,9,10]].values,df.iloc[:, 4].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [16]:
```
classifier.fit(X_train,y_train)
y_predict = classifier.predict(X_test)

#print the accuracy
print("Accuracy:",accuracy_score(y_predict,y_test))
```

```
Accuracy: 0.6475335816683598
```

In [17]:
```
#print the confusion matrix
conf_mat = confusion_matrix(y_predict,y_test)
print(conf_mat)
```

```
[[6528 4100]
 [2145 4945]]
```

Comment on the difference in accuracy between both models.

The accelerometer provides changes in device's velocity along 3 axes, which is a crucial piece of information about how one moves his hand. And a gyroscope delivers the rate at which a device rotates around a spatial axis which carries probably fewer insights for distinguishing running and walking activities.

Using our model, the accuracy score obtained using 'Acceleration value' as 'predictor' is 95%. And the accuracy score obtained using 'Gyro value' as 'predictor' is 64%.

From the above results, we could cleary say accelerometer data provides better results when distinguishing 'run or walk' compared to Gyroscope data.