In [1]:
```
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
# Importing Classifier Modules
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import GradientBoostingClassifier

import warnings
warnings.filterwarnings('ignore')

import os
```

In [2]:
```
df=pd.read_csv('dataset.csv')
```

In [3]:
```
new_df = pd.DataFrame()
new_df = df.copy()
new_df
```

Out[3]:

	date	time	username	wrist	activity	acceleration_x	acceleration_y	accelera
0	2017-06-30	13:51:15:847724020	viktor	0	0	0.2650	-0.7814	-
1	2017-06-30	13:51:16:246945023	viktor	0	0	0.6722	-1.1233	-
2	2017-06-30	13:51:16:446233987	viktor	0	0	0.4399	-1.4817	
3	2017-06-30	13:51:16:646117985	viktor	0	0	0.3031	-0.8125	
4	2017-06-30	13:51:16:846738994	viktor	0	0	0.4814	-0.9312	
...	...	...	...	...	...	...	...	
88583	2017-07-09	20:9:15:317911028	viktor	0	0	0.3084	-0.8376	-
88584	2017-07-09	20:9:15:517889022	viktor	0	0	0.4977	-1.0027	-
88585	2017-07-09	20:9:15:717828989	viktor	0	0	0.4587	-1.1780	-
88586	2017-07-09	20:9:15:917932987	viktor	0	0	0.2590	-0.8582	-
88587	2017-07-09	20:9:16:117410004	viktor	0	0	0.3140	-0.8008	-

88588 rows × 11 columns

In [4]:
```
new_df = new_df.drop("date", axis = 1)
new_df = new_df.drop("time", axis = 1)
new_df = new_df.drop("username", axis=1)
```

In [5]:
```
new_df.head()
```

Out[5]:

	wrist	activity	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z
0	0	0	0.2650	-0.7814	-0.0076	-0.0590	0.0325	-2.9296
1	0	0	0.6722	-1.1233	-0.2344	-0.1757	0.0208	0.1269
2	0	0	0.4399	-1.4817	0.0722	-0.9105	0.1063	-2.4367
3	0	0	0.3031	-0.8125	0.0888	0.1199	-0.4099	-2.9336
4	0	0	0.4814	-0.9312	0.0359	0.0527	0.4379	2.4922

In [6]:
```
new_df.shape
```

Out[6]:
```
(88588, 8)
```

In [7]:
```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88588 entries, 0 to 88587
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   wrist           88588 non-null  int64  
 1   activity        88588 non-null  int64  
 2   acceleration_x  88588 non-null  float64
 3   acceleration_y  88588 non-null  float64
 4   acceleration_z  88588 non-null  float64
 5   gyro_x          88588 non-null  float64
 6   gyro_y          88588 non-null  float64
 7   gyro_z          88588 non-null  float64
dtypes: float64(6), int64(2)
memory usage: 5.4 MB
```

In [8]:
```
new_df = new_df.drop("activity", axis=1)
```

In [9]:
```
new_df["label"] = df["activity"]
```

In [10]:
```
new_df.head()
```

Out[10]:

	wrist	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z	label
0	0	0.2650	-0.7814	-0.0076	-0.0590	0.0325	-2.9296	0
1	0	0.6722	-1.1233	-0.2344	-0.1757	0.0208	0.1269	0
2	0	0.4399	-1.4817	0.0722	-0.9105	0.1063	-2.4367	0
3	0	0.3031	-0.8125	0.0888	0.1199	-0.4099	-2.9336	0
4	0	0.4814	-0.9312	0.0359	0.0527	0.4379	2.4922	0

In [11]:
```
from sklearn.model_selection import train_test_split

y = new_df["label"]
x = new_df.iloc[:,0:7]

x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    train_size=0.7,
                                                    random_state=42)

print(f"Train labels:\n{y_train}")
print(f"Test labels:\n{y_test}")
```

```
Train labels:
77386    0
61379    1
54799    1
12789    1
33035    0
        ..
6265     0
54886    0
76820    0
860      1
15795    1
Name: label, Length: 62011, dtype: int64
Test labels:
57800    1
53690    1
75294    1
16113    1
88456    0
        ..
63775    1
10554    1
41523    0
54561    1
46039    0
Name: label, Length: 26577, dtype: int64
```

In [12]:
```
df_a = x_train
df_b = y_train

training_data = pd.concat([df_a,df_b],axis = 1, join = "inner")

training_data
```

Out[12]:

	wrist	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z	label
77386	0	0.1248	-0.9639	-0.1289	0.1370	3.0972	0.0588	0
61379	0	0.6582	0.3590	-0.8375	-0.6977	-0.6435	-3.6945	1
54799	1	-0.2007	-0.1754	0.1220	0.4007	-1.0399	-0.7537	1
12789	1	-2.0388	-0.1770	-0.2392	1.9371	0.8875	1.7828	1
33035	1	-0.1635	-0.4766	-0.0746	-0.9139	-0.6624	2.0334	0
...	...	...	...	...	...	...	...	...
6265	1	-0.1714	-1.0067	-0.2231	-1.4891	0.1767	1.7018	0
54886	0	0.4980	-1.0698	-0.1829	1.0147	1.0042	3.2428	0
76820	0	0.3459	-0.8581	-0.0603	-0.6830	-0.0735	-1.3448	0
860	0	-0.4821	-0.5633	0.0581	0.4138	0.5662	0.6665	1
15795	1	-0.4902	0.4210	-0.1457	2.1190	-0.1988	4.0985	1

62011 rows × 8 columns

In [13]:
```
testing_data = pd.concat([x_test,y_test], axis = 1, join = "inner")

testing_data
```

Out[13]:

	wrist	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z	label
57800	0	1.6815	-0.4641	-0.9647	0.2961	-1.5709	-1.6958	1
53690	1	-0.1132	0.4044	-0.2144	1.7535	-0.7964	3.5975	1
		2.2226	-1.8281	-2.2184	-0.1521	-0.3216	2.7630	1

	wrist	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z	label
16113	1	0.0571	0.5462	-0.1377	0.7485	0.0477	2.0552	1
88456	0	0.3597	-1.2908	-0.3557	-0.8036	-0.7982	-1.3781	0
...	...	...	...	...	...	...	...	...
63775	0	0.7948	-1.0409	-0.5441	1.2226	-0.7278	2.9079	1
10554	1	-2.5426	-0.1672	-0.9576	-1.9312	0.2586	-2.1163	1
41523	1	-0.3327	-0.6975	-0.1221	0.5792	-0.0855	-1.1447	0
54561	1	-1.5603	-0.1332	-1.0948	-2.3559	-0.2329	-0.3394	1
46039	0	0.3243	-1.2243	-0.1308	1.0055	-1.2581	1.6850	0

26577 rows × 8 columns

In [14]:
```
#training_data.to_csv(os.getcwd() + "/Desktop/training data.csv", index = False)
#testing_data.to_csv(os.getcwd() + "/Desktop/testing data.csv", index = False)
```

In [16]:
```
train_df=pd.read_csv('training data.csv')
train_df.head()
nt_df = train_df.copy()
```

In [18]:
```
# split training and testing data in the training set for model training process
Y = nt_df["label"]
X = nt_df.iloc[:,0:7]
train_x, test_x, train_y, test_y = train_test_split(X, Y,
                                                    train_size=0.7,
                                                    random_state=42)
```

In [19]:
```
#Training models

def fit_model (model):
    classifier = model()  #train with default model parameters
    classifier.fit(train_x, train_y)
    print("training accuracy is:",classifier.score(train_x, train_y))
    print("testing accuracy is:",classifier.score(test_x, test_y))
    return classifier
```

In [20]:
```
# the fit_model function will return the mean accuracy of given test data and labels

# calculate precision, recall, fscore, and support score for the model
from sklearn.metrics import precision_recall_fscore_support

def prfs (trained_model):
    pred_label = trained_model.predict(test_x)
    print("When positive class refers to \"running\", the precision, recall, f_measure an
          precision_recall_fscore_support(test_y, pred_label, average = "binary"))
```

In [21]:
```
#SVC
svc = fit_model(SVC)
prfs(svc)
```

```
training accuracy is: 0.9884350450388186
testing accuracy is: 0.9879595785852505
When positive class refers to "running", the precision, recall, f_measure and support for
the model is : (0.9904228989562036, 0.9855444908448442, 0.9879776728209533, None)
```

In [22]:
```
#decision tree
dt = fit_model(DecisionTreeClassifier)
prfs(dt)
```

```
training accuracy is: 1.0
testing accuracy is: 0.9836056761986669
When positive class refers to "running", the precision, recall, f_measure and support for
the model is : (0.9850730240549829, 0.9822250776314381, 0.9836469894375636, None)
```

In [23]:
```
#random forest
rf = fit_model(RandomForestClassifier)
prfs(rf)
```

```
training accuracy is: 1.0
testing accuracy is: 0.9903246613631477
When positive class refers to "running", the precision, recall, f_measure and support for
the model is : (0.9895243185462319, 0.9912196166613128, 0.9903712421097678, None)
```

In [24]:
```
#logistic regression

lr = fit_model(LogisticRegression)
prfs(lr)
```

```
training accuracy is: 0.8588937268182552
testing accuracy is: 0.8600301010535368
When positive class refers to "running", the precision, recall, f_measure and support for
the model is : (0.8956644342615439, 0.8162544169611308, 0.8541176470588235, None)
```

In [25]:
```
#gradient boosting model
gb = fit_model(GradientBoostingClassifier)
prfs(gb)
```

```
training accuracy is: 0.9854170986246458
testing accuracy is: 0.9835519243173511
When positive class refers to "running", the precision, recall, f_measure and support for
the model is : (0.9880064829821718, 0.9791198201092194, 0.9835430784123911, None)
```

In [26]:
```
#stochastic gradient decient model
sgd = fit_model(SGDClassifier)
prfs(sgd)
```

```
training accuracy is: 0.8632478632478633
testing accuracy is: 0.8641152440335411
When positive class refers to "running", the precision, recall, f_measure and support for
the model is : (0.9305854090276899, 0.7880929435699754, 0.8534322820037105, None)
```

In [27]:
```
#perceptron classifier
perceptron = fit_model(Perceptron)
prfs(perceptron)
```

```
training accuracy is: 0.8499090008523971
testing accuracy is: 0.8485271984519458
When positive class refers to "running", the precision, recall, f_measure and support for
the model is : (0.8971860153490072, 0.7886283327979441, 0.839411898791885, None)
```

In [28]:
```
#naive baysian classifier:
nb = fit_model(GaussianNB)
prfs(nb)
```

```
training accuracy is: 0.9560669938028429
testing accuracy is: 0.9566759836594281
When positive class refers to "running", the precision, recall, f_measure and support for
the model is : (0.9896706071387582, 0.9233322625548774, 0.9553512076224241, None)
```

In [ ]:
```
#random forest is the best with best accuracy score and f_score; all models don't have ov
```

In [ ]:
```

```