–SPL-II-Assignment-I –

December 5, 2023

1 Mobile Sensor-based Activity Classification using Naive Bayes Algorithm

[ ]:

```

```

1.Load the kinematics dataset as measured on mobile sensors from the file "run_or_walk.csv". List out the columns in the dataset.

[1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:

```
data = pd.read_csv("C:\\Users\\DELL\\Documents\\Supervised Learning␣
↪Today\\09)Supervised Learning - II\\run_or_walk.csv")
data.head()
```

[2]:

```
        date                time username  wrist  activity  acceleration_x  \
0  2017-6-30  13:51:15:847724020   viktor      0         0          0.2650
1  2017-6-30  13:51:16:246945023   viktor      0         0          0.6722
2  2017-6-30  13:51:16:446233987   viktor      0         0          0.4399
3  2017-6-30  13:51:16:646117985   viktor      0         0          0.3031
4  2017-6-30  13:51:16:846738994   viktor      0         0          0.4814

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

[3]:

```
data.columns
```

[3]:

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

[ ]:

```

```

2.Let the target variable 'y' be the activity and assign all the columns after it to 'x'.

[4]:
```
X = data.iloc[:, 5:]
Y = data["activity"]
```

3.Using Scikit-learn fit a Gaussian Naive Bayes model and observe the accuracy.Generate a classification report using scikit learn.

[5]:
```
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

[6]:
```
train_x, test_x, train_y, test_y = train_test_split(X, Y, random_state=10,␣
↪test_size=0.30)

g_model = GaussianNB()
g_model.fit(train_x, train_y)

predicted_values = g_model.predict(test_x)

print("\nAccuracy Score\n")
print(metrics.accuracy_score(predicted_values, test_y))
```

```
Accuracy Score

0.9580840576438274
```

[7]:
```
print("\nClassification Score\n")
print(metrics.classification_report(predicted_values, test_y))
```

```
Classification Score

              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
```

[ ]:
```

```

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.

[8]:
```python
# Accelerations as Independent variables
X_A = data.iloc[:, 5:8]
Y_A = data["activity"]

train_x_a, test_x_a, train_y_a, test_y_a = train_test_split(X_A, Y_A,␣
↪random_state=10, test_size=0.30)

g_model.fit(train_x_a, train_y_a)
predicted_values_a = g_model.predict(test_x_a)

print("\nAccuracy Score\n")
print(metrics.accuracy_score(predicted_values_a, test_y_a))
```

```
Accuracy Score

0.958648455431388
```

[9]:
```python
print("\nClassification Score\n")
print(metrics.classification_report(predicted_values_a, test_y_a))
```

```
Classification Score

              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
```

[ ]:
```python

```

[10]:
```python
# Gyro as Independent variables
X_G = data.iloc[:, 8:]
Y_G = data["activity"]

train_x_g, test_x_g, train_y_g, test_y_g = train_test_split(X_G, Y_G,␣
↪random_state=10, test_size=0.30)

g_model.fit(train_x_g, train_y_g)
predicted_values_g = g_model.predict(test_x_g)

print("\nAccuracy Score\n")
print(metrics.accuracy_score(predicted_values_g, test_y_g))
```

```
Accuracy Score

0.6486811905030666
```

[11]:
```python
print("\nClassification Score\n")
print(metrics.classification_report(predicted_values_g, test_y_g))
```

```
Classification Score

              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
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: