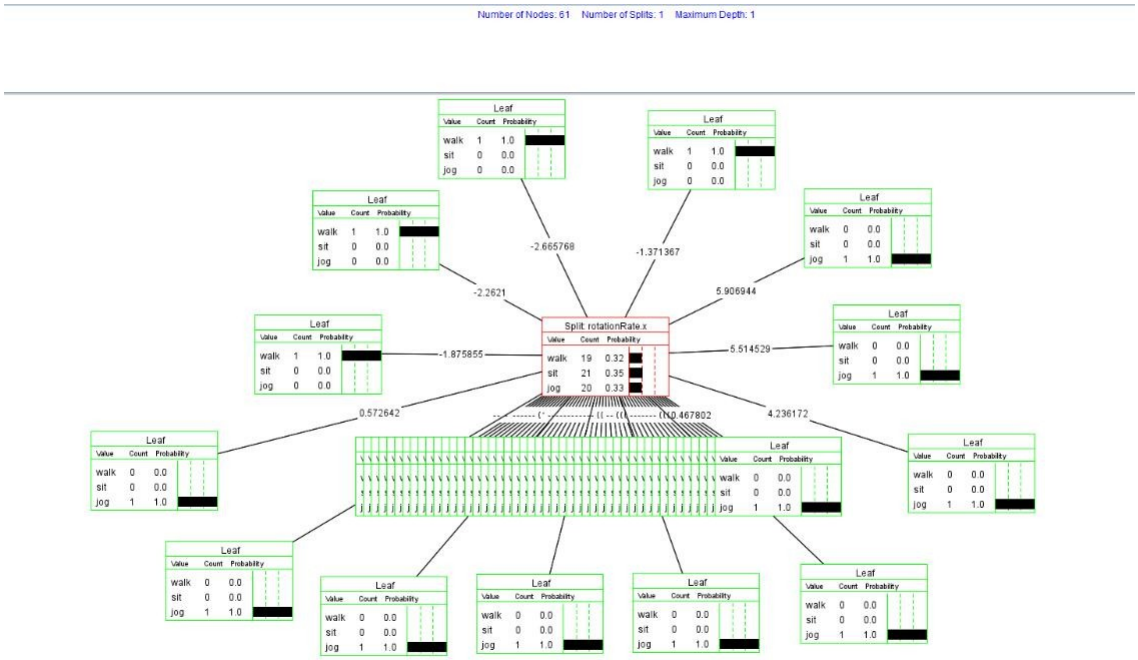
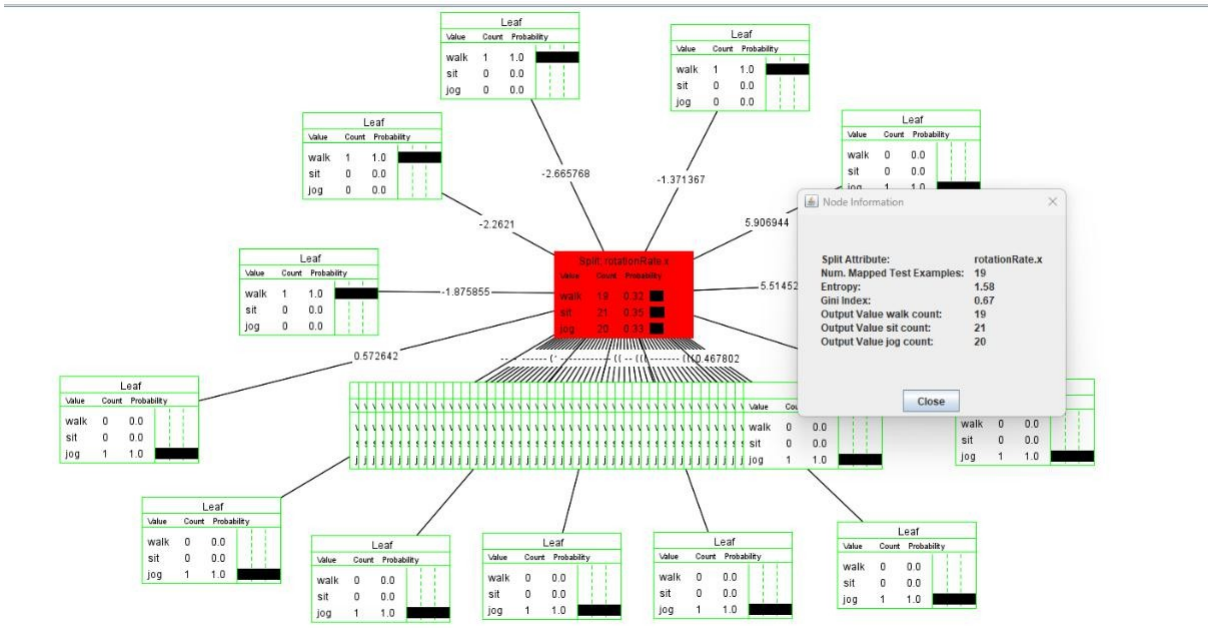
DT Human Activity Assignment

Raphael J. Malims 663762



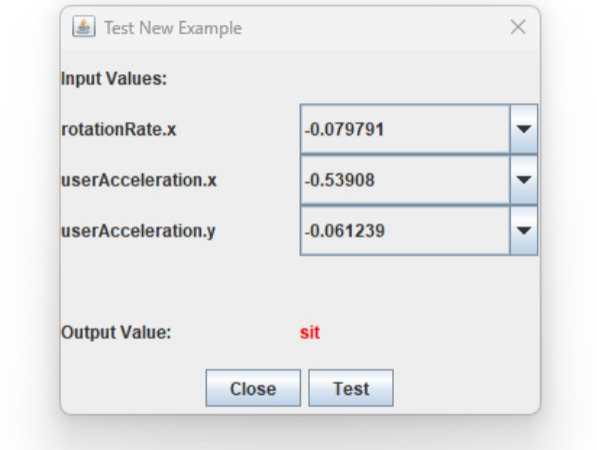
Explanation:

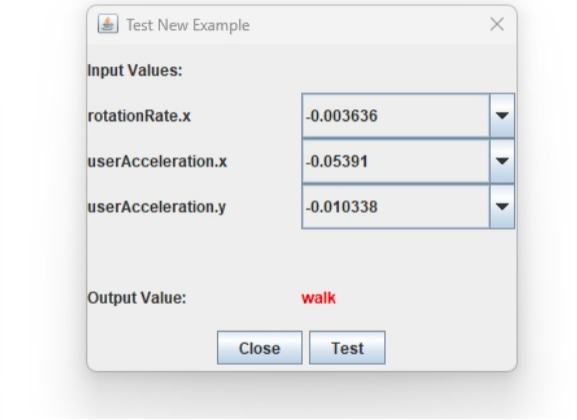
I employed the DTree Java App to analyze device motion, specifically targeting userAcceleration.x, userAcceleration.y, rotationRate.x, and class columns. The decision tree was designed to distinguish between sitting, walking, and jogging activities. The dataset comprised 78 rows, split into 60 for training and 19 for testing.

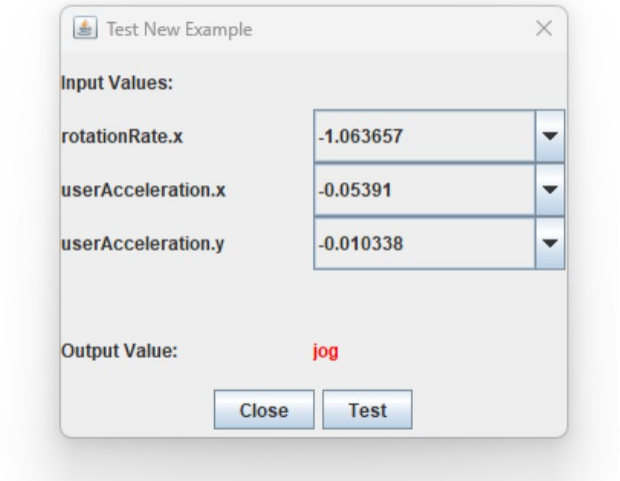


Explanations:
In my examination, I underscored the significance of rotationRate.x within the decision tree. The noteworthy entropy of 1.58 emphasized its efficiency in accurately categorizing the data. This minimal entropy indicated the pivotal role of rotationRate.x in establishing clear groupings for precise classification in my scenario.

Data Exploration:







Explanation: The above is a brief glimpse into the exploration of different data points to show that data does vary.

Code:

Using the provided data, I constructed a decision tree, visualized it, conducted testing, and assessed its accuracy, resulting in a 72% precision rate. The code to how I achieved this is below:

```
import pandas as pd

data = pd.read_csv('/content/data.csv', header='infer')

data
```

```
    rotationRate.x  userAcceleration.x  userAcceleration.y  class
0        -2.548994            0.391979            0.913534   walk
1        -2.665768            0.611504            0.886597   walk
2        -2.854574            0.636739            0.767762   walk
3        -2.516332            0.581910            0.588902   walk
4        -2.262100            0.164042            0.029239   walk
..             ...                 ...                 ...    ...
73        2.783017           -0.953833           -0.061239    jog
74        4.236172           -0.579825            2.112855    jog
75        5.514529           -0.846420            2.751029    jog
76        5.906944           -0.541555            0.921610    jog
77        4.903476            0.001728            0.208539    jog

[78 rows x 4 columns]
```

```
data.info()
```

```
<class
'pandas.core.frame.DataFrame'>
RangeIndex: 78 entries, 0 to 77
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
                       -
 0   rotationRate.x    78 non-null     float6
                                       4
 1   userAcceleration.  78 non-null    float6
     x                                 4
 2   userAcceleration.  78 non-null    float6
     y                                 4
 3   class             78 non-null     object
dtypes: float64(3), object(1)
memory usage: 2.6+ KB
```

```
data['class']
```

```
0     walk
1     walk
2     walk
3     walk
4     walk
      ...
73     jog
74     jog
```

```
75    jog
76    jog
77    jog
Name: class, Length: 78, dtype: object
```

```
from sklearn import tree
```

```
y = data['class']
y
```

```
0     walk
1     walk
2     walk
3     walk
4     walk
      ...
73     jog
74     jog
75     jog
76     jog
77     jog
Name: class, Length: 78, dtype: object
```

```
X = data.drop(['class'], axis = 1)
X
```

```
    rotationRate.x  userAcceleration.x  userAcceleration.
                                                        y
0        -2.548994           0.391979          0.913534
1        -2.665768           0.611504          0.886597
2        -2.854574           0.636739          0.767762
3        -2.516332           0.581910          0.588902
4        -2.262100           0.164042          0.029239
..             ...                ...               ...
73        2.783017          -0.953833         -0.061239
74        4.236172          -0.579825          2.112855
75        5.514529          -0.846420          2.751029
76        5.906944          -0.541555          0.921610
77        4.903476           0.001728          0.208539

[78 rows x 3 columns]
```

```
clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth =
3)
```

```
clf
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

```
clf = clf.fit(X,y)
clf
```

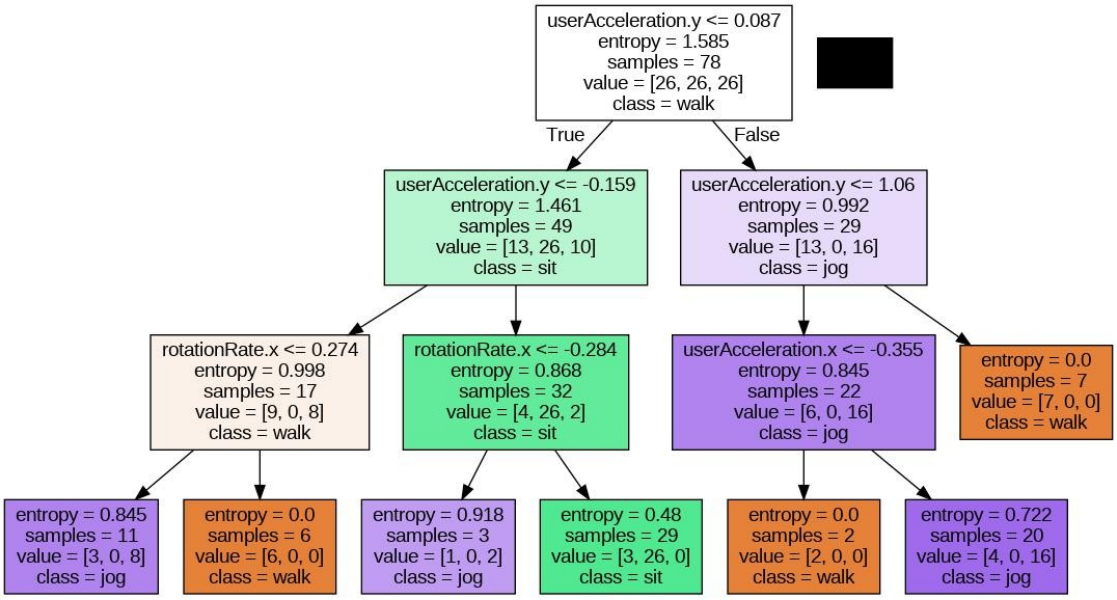
```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

```
import pydotplus
```

```
from IPython.display import Image
```

```
dot_data = tree.export_graphviz(clf, feature_names = X.columns,
class_names = ['walk', 'sit', 'jog'], filled = True, out_file =
None)
```

```
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```




```
testData=pd.read_csv('/content/testing2.csv',header='infer')
```

```
testData
```

```
    rotationRate.x  userAcceleration.x  userAcceleration.y  class
0         0.362281           -0.093681           -0.556827   walk
1         0.035447           -0.132940           -0.578839   walk
2        -0.277767           -0.140426           -0.551390   walk
3        -0.658007           -0.138939           -0.447676   walk
4        -1.088610           -0.134953           -0.351983   walk
..             ...                 ...                 ...    ...
56       -0.375877            1.438368            0.284167    jog
57       -5.429132            3.572135           -1.175655    jog
58      -10.613199            2.616796           -1.676538    jog
59       -0.175480            0.746894           -0.416966    jog
60        2.228408           -0.085134            0.587051    jog

[61 rows x 4 columns]
```

```
testData = pd.DataFrame(testData, columns = data.columns)
```

```
testData
```

```
    rotationRate.x  userAcceleration.x  userAcceleration.y  class
0         0.362281           -0.093681           -0.556827   walk
1         0.035447           -0.132940           -0.578839   walk
2        -0.277767           -0.140426           -0.551390   walk
3        -0.658007           -0.138939           -0.447676   walk
4        -1.088610           -0.134953           -0.351983   walk
..             ...                 ...                 ...    ...
56       -0.375877            1.438368            0.284167    jog
57       -5.429132            3.572135           -1.175655    jog
58      -10.613199            2.616796           -1.676538    jog
59       -0.175480            0.746894           -0.416966    jog
```

```
60       2.228408         -0.085134          0.587051  jog

[61 rows x 4 columns]
```

```
testY = testData['class']
testX = testData.drop(['class'], axis = 1)

predY = clf.predict(testX)

predY
```

```
array(['jog', 'walk', 'walk', 'walk', 'walk', 'walk', 'walk',
       'walk', 'walk', 'walk', 'walk', 'sit', 'walk', 'walk', 'sit',
       'walk', 'walk', 'walk', 'walk', 'walk', 'walk', 'sit', 'sit',
       'sit', 'sit', 'sit', 'sit', 'sit', 'sit', 'sit', 'sit',
       'sit', 'sit', 'sit', 'sit', 'sit', 'sit', 'sit', 'sit',
       'sit', 'sit', 'walk', 'jog', 'jog', 'walk', 'walk', 'walk',
       'walk', 'walk', 'walk', 'walk', 'jog', 'jog', 'jog', 'jog',
       'walk', 'walk', 'walk', 'walk', 'walk', 'walk'],
      dtype=object)
```

```
predictions = pd.concat([testData['class'], pd.Series(predY, name =
'Predicted Class')], axis = 1)
predictions
```

```
       class Predicted
                 Class
0   walk           jog
1   walk          walk
2   walk          walk
3   walk          walk
4   walk          walk
..   ...           ...
56   jog          walk
57   jog          walk
58   jog          walk
59   jog          walk
60   jog          walk

[61 rows x 2 columns]
```

```
from sklearn.metrics import

accuracy_score

accuracy_score(testY,predY)
```

```
0.7213114754098361
```

```
maxDepth =
[2,3,4,5,6,7,8,9,10,15,20,25,30,35,40,45,50]
maxDepth
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50]
```

```
import numpy as np
trainAcc =np.zeros(len(maxDepth))
trainAcc
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.])
```

```
testAcc = np.zeros(len(maxDepth))
testAcc
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0.])
```

```
index = 0
for depth in maxDepth:
  clf = tree.DecisionTreeClassifier(max_depth = depth)
  clf = clf.fit(X,y)
  Y_predTrain = clf.predict(X)
  Y_predTest = clf.predict(testX)
  trainAcc[index] = accuracy_score(y,
  Y_predTrain) testAcc[index] =
  accuracy_score(testY, Y_predTest) index +=1
```

```
trainAcc
```

```
array([0.74358974, 0.87179487, 0.93589744, 0.97435897, 0.97435897,
       0.98717949 0.98717949  1.       , 1.       , 1.       ,
                , ,
       1.       , 1.       , 1.       , 1.       , 1.       ,
       1.       , 1.       ])
```

```
testAcc
```

```
         array([0.63934426, 0.72131148, 0.75409836, 0.7704918 ,
    0.86885246, 0.86885246, 0.86885246, 0.90163934, 0.7704918 ,
    0.90163934, 0.80327869, 0.8852459 , 0.8852459 , 0.8852459 ,
                                                    0.80327869,
      0.90163934, 0.86885246])
```

```
import matplotlib.pyplot as plt
plt.plot(maxDepth, trainAcc, 'ro-', maxDepth, testAcc, 'bv--')
plt.legend(['Training Accuracy', 'Test Accuracy'])
plt.xlabel('Max Depth')
plt.ylabel('Accuracy')
```

```
Text(0, 0.5, 'Accuracy')
```