Assignment Project Report

Human Activity Recognition from Smart Phone Data

Name : Meghna Maan

Course : AI & ML (Batch 4)

Given Question

Perform activity recognition on the dataset using a hidden markov model. Then perform the same task using a different classification algorithm (logistic regression/decision tree) of your choice and compare the performance of the two algorithms

Prerequisites

1. Software:

   Python 3

2. Tools:

   - Numpy
   - Pandas
   - Matplotlib
   - Opencv
   - Pandas
   - Seaborn

- Warning

Datase: https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones

Methods Used

Recognizing human activities from temporal streams of sensory data observations is a very important task on a wide variety of applications in context recognition. Human activities are hierarchical in nature, i.e. the complex activities can be decomposed to several simpler ones. Human activity recognition is the problem of classifying sequences of accelerometer data recorded by pre-installed sensors in smart phones into known well-defined movements to make it ready for predictive modelling.

Implementation

1. Loading Libraries and Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

2. Training and printing the data set

```
train = pd.read_csv("/Users/prahaladsinghmaan/Desktop/train.csv")
test = pd.read_csv('/Users/prahaladsinghmaan/Desktop/test.csv')
```

```
train.head()
```

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fl
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	

3. Checking for duplicate or null values

```
print('Number of duplicates in train : ',sum(train.duplicated()))
print('Number of duplicates in test : ', sum(test.duplicated()))
```

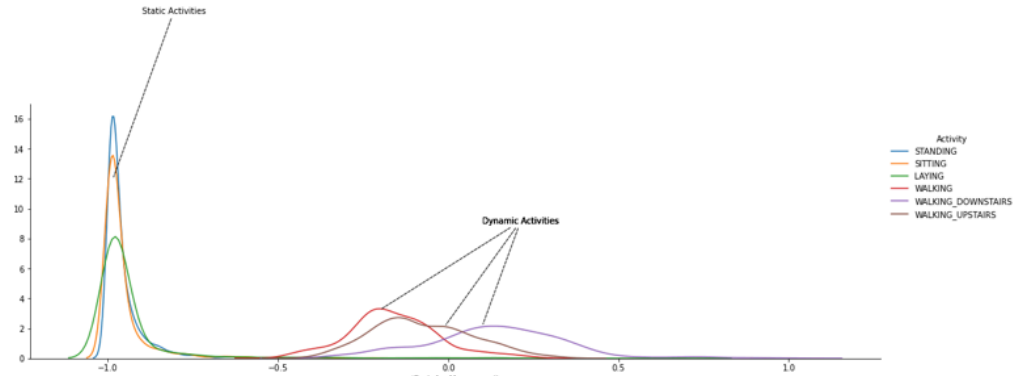
```
Number of duplicates in train :  0
Number of duplicates in test :  0
```

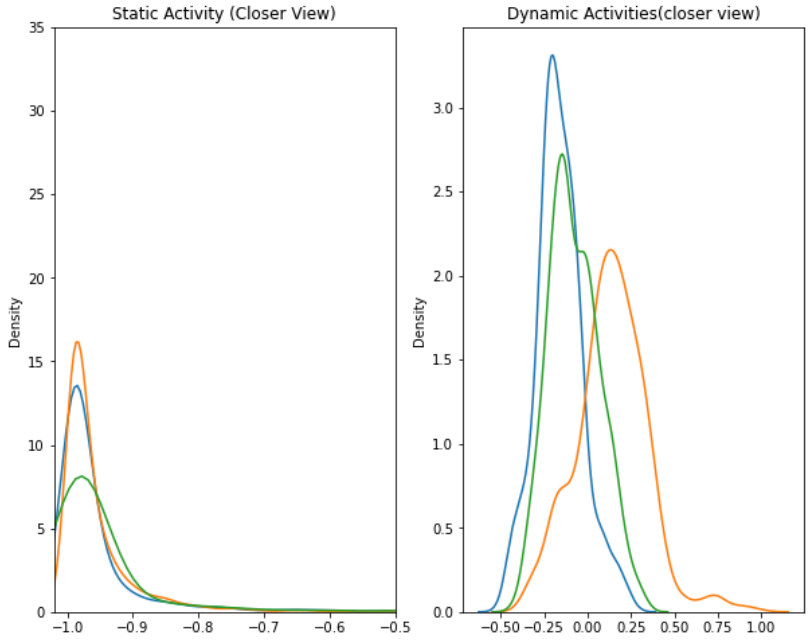
```
print('Total number of null values in train:',train.isna().values.sum())
print('Total number of null values in test:',test.isna().values.sum())
```

```
Total number of null values in train: 0
Total number of null values in test: 0
```

4. Visualizing the data

```
facetgrid = sns.FacetGrid(train, hue='Activity', height=5,aspect=3)
facetgrid.map(sns.distplot,'tBodyAccMag-mean()', hist=False).add_legend()
plt.annotate("Static Activities", xy=(-.996,21), xytext=(-0.9, 23),arrowprops={'arrowstyle':
plt.annotate("Static Activities", xy=(-.990,26), xytext=(-0.9, 23),arrowprops={'arrowstyle':
plt.annotate("Static Activities", xy=(-0.985,12), xytext=(-0.9, 23),arrowprops={'arrowstyle'
plt.annotate("Dynamic Activities", xy=(-0.2,3.25), xytext=(0.1, 9),arrowprops={'arrowstyle':
plt.annotate("Dynamic Activities", xy=(0.1,2.18), xytext=(0.1, 9),arrowprops={'arrowstyle':
plt.annotate("Dynamic Activities", xy=(-0.01,2.15), xytext=(0.1, 9),arrowprops={'arrowstyle'
```





5. Implementing required classification algorithms

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score,classification_report
```

```
parameters = {'C':np.arange(10,61,10), 'penalty':['l2','l1']}
lr_classifier = LogisticRegression()
lr_classifier_rs = RandomizedSearchCV(lr_classifier,param_distributions=parameters,cv=5,random_state=42)
lr_classifier_rs.fit(X_train, y_train)
y_pred = lr_classifier_rs.predict(X_test)
```

6. Decision Tree

```
dt_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy using Decision tree : ", dt_accuracy)
```

```
Accuracy using Decision tree :  0.8696979979640312
```

7. Hidden Markov Model

```
from hmmlearn import hmm
model = hmm.GaussianHMM(n_components=3, covariance_type="full", n_iter=100)
```

```
model.fit(X_train)
```

```
GaussianHMM(covariance_type='full', n_components=3, n_iter=100)
```

```
y_pred_hmm = model.predict(X_test)
```

```
np.unique(y_pred_hmm)
```

```
array([0, 1, 2], dtype=int32)
```