

PROJECT REPORT

Assignment Project Report: Human Activity Recognition from Smart Phone Data

NAME: RISHI RAJ SINGH

AI AND ML B-4

Problem Statement:

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:

- Software: Python 3

Tools:

- Pandas
- Numpy
- Matplotlib
- Sklearn
- Seaborn

Method 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.

Dataset Used :

Human Activity Recognition with Smartphones

Implementation:

1. Load all required libraries

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

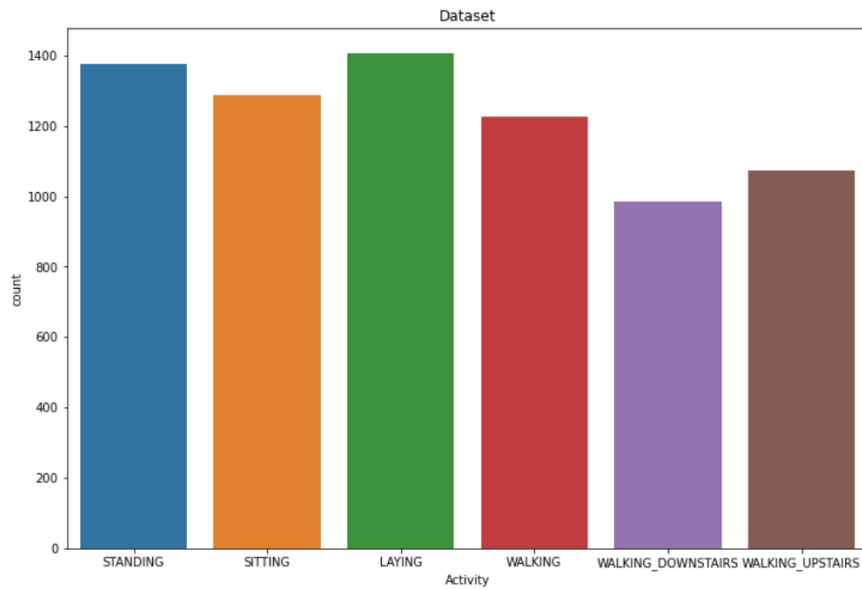
2. Loading Dataset

```
In [3]: train=pd.read_csv("train.csv")
test=pd.read_csv("test.csv")
```

```
In [4]: train.head(10)
```

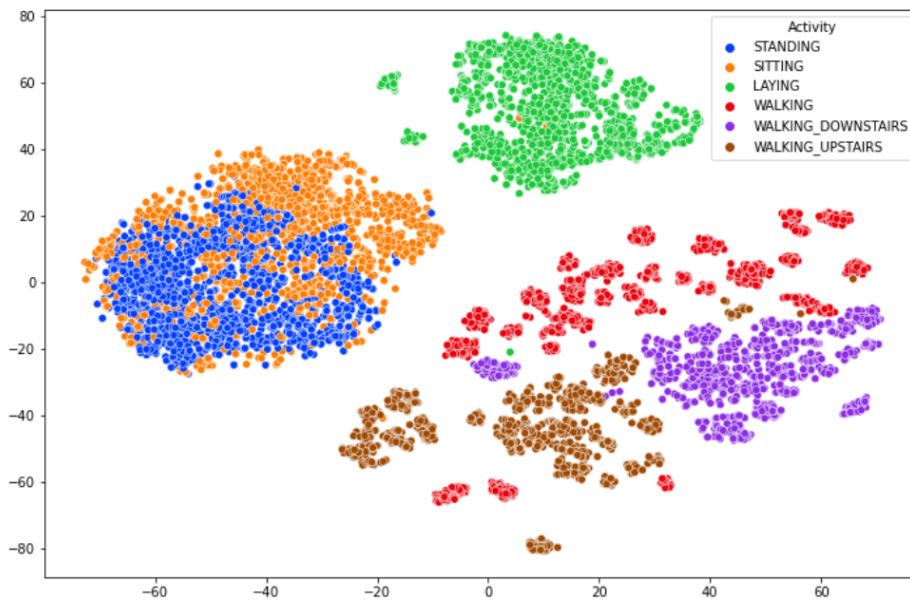
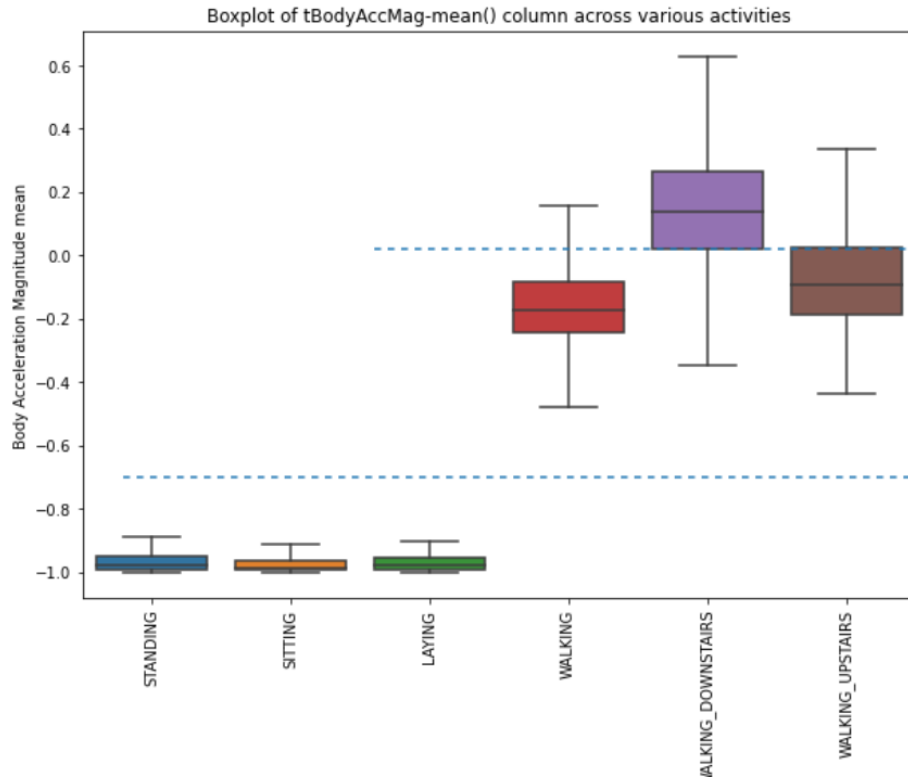
3. Visualization of data

```
#checking if there is imbalance in data
plt.figure(figsize=(12,8))
sns.countplot(train["Activity"])
plt.title("Dataset")
plt.show()
```



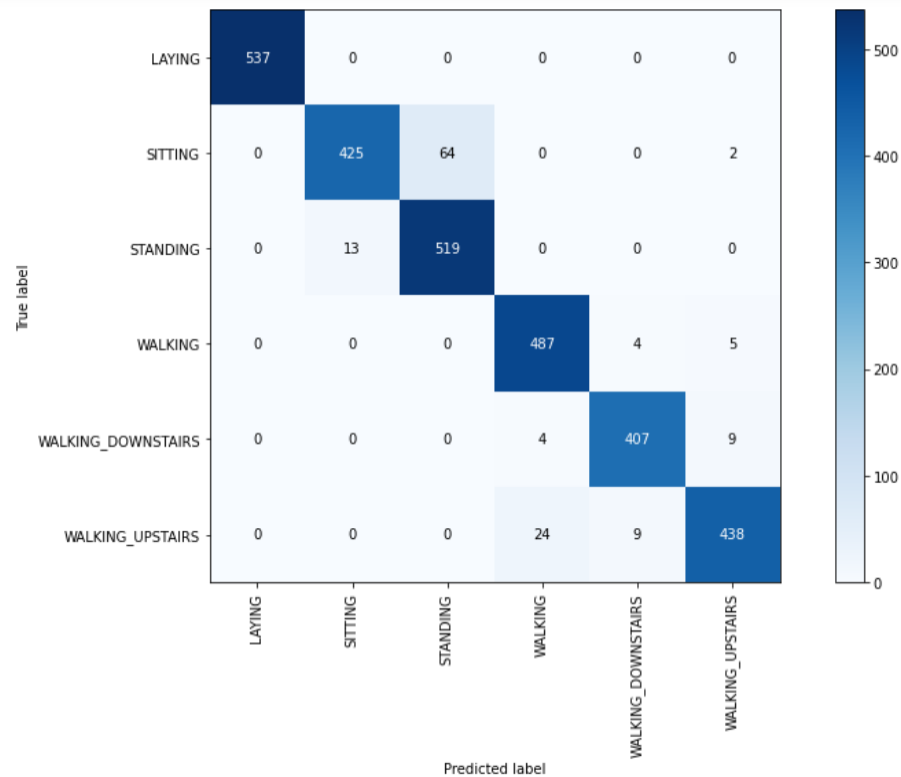
```
In [15]: plt.figure(figsize=(10,7))
sns.boxplot(x='Activity', y='tBodyAccMag-mean()',data=train, showfliers=False)
plt.ylabel('Body Acceleration Magnitude mean')
plt.title("Boxplot of tBodyAccMag-mean() column across various activities")
plt.axhline(y=-0.7, xmin=0.05,dashes=(3,3))
plt.axhline(y=0.020, xmin=0.35, dashes=(3,3))
plt.xticks(rotation=90)
```

```
Out[15]: (array([0, 1, 2, 3, 4, 5]),
```



4. Confusion Matrix

```
In [40]: # function to plot confusion matrix
def plot_confusion_matrix(cm, labels):
    fig, ax = plt.subplots(figsize=(12,8)) # for plotting confusion matrix as image
    im = ax.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    ax.figure.colorbar(im, ax=ax)
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           xticklabels=labels, yticklabels=labels,
           ylabel='True label',
           xlabel='Predicted label')
    plt.xticks(rotation = 90)
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, int(cm[i, j]), ha="center", va="center", color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
```



5. Random Search


```
In [42]: #function to get best random search attributes
def get_best_randomsearch_results(model):
    print("Best estimator : ", model.best_estimator_)
    print("Best set of parameters : ", model.best_params_)
    print("Best score : ", model.best_score_)
```

```
In [43]: get_best_randomsearch_results(lr_classifier_rs)
```

```
Best estimator : LogisticRegression(C=20)
Best set of parameters : {'penalty': 'l2', 'C': 20}
Best score : 0.9311818976400893
```

6.HMM

```
In [44]:  from hmmlearn import hmm  
model = hmm.GaussianHMM(n_components=2, covariance_type="full", n_iter=100)
```

```
In [45]:  model.fit(X_train)
```

```
Out[45]: GaussianHMM(covariance_type='full', n_components=2, n_iter=100)
```
