Assignment Project Report

Human Activity Recognition from Smart Phone Data

Name: Pavan Tiwari

Course: AI and ML
(Batch 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 (Use anaconda as your python distributor as well)

  - Tools:
    - Pandas
    - Numpy
    - Matplotlib
    - Sklearn
    - Seaborn

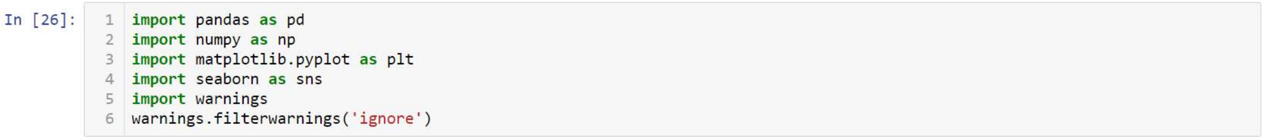
  - Dataset Link: Human Activity Recognition with Smartphones
    https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones

- **Method Used**

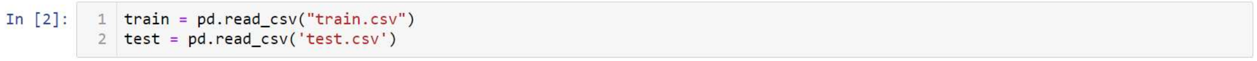
Recognizing human activities from temporal streams of sensory data observations is a very important task on a wide variety of applications in context recognition. Human activities are hierarchical in nature, i.e. the complex activities can be decomposed to several simpler ones. Human activity recognition 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.

- **Implementation:**

1. Load all required libraries



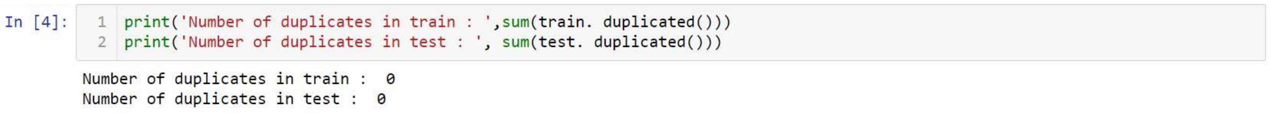
```
In [26]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         import warnings
         warnings.filterwarnings('ignore')
```



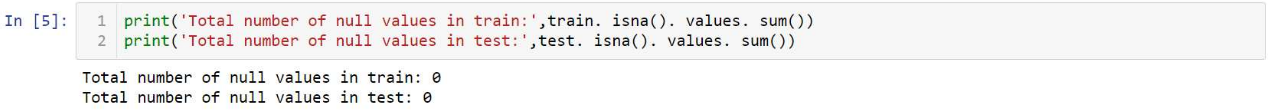
```
In [2]:  train = pd.read_csv("train.csv")
          test = pd.read_csv('test.csv')
```

```
In [3]:  train.head()
```

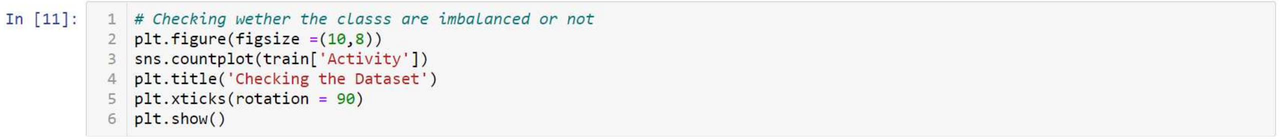
2. Visualization of data



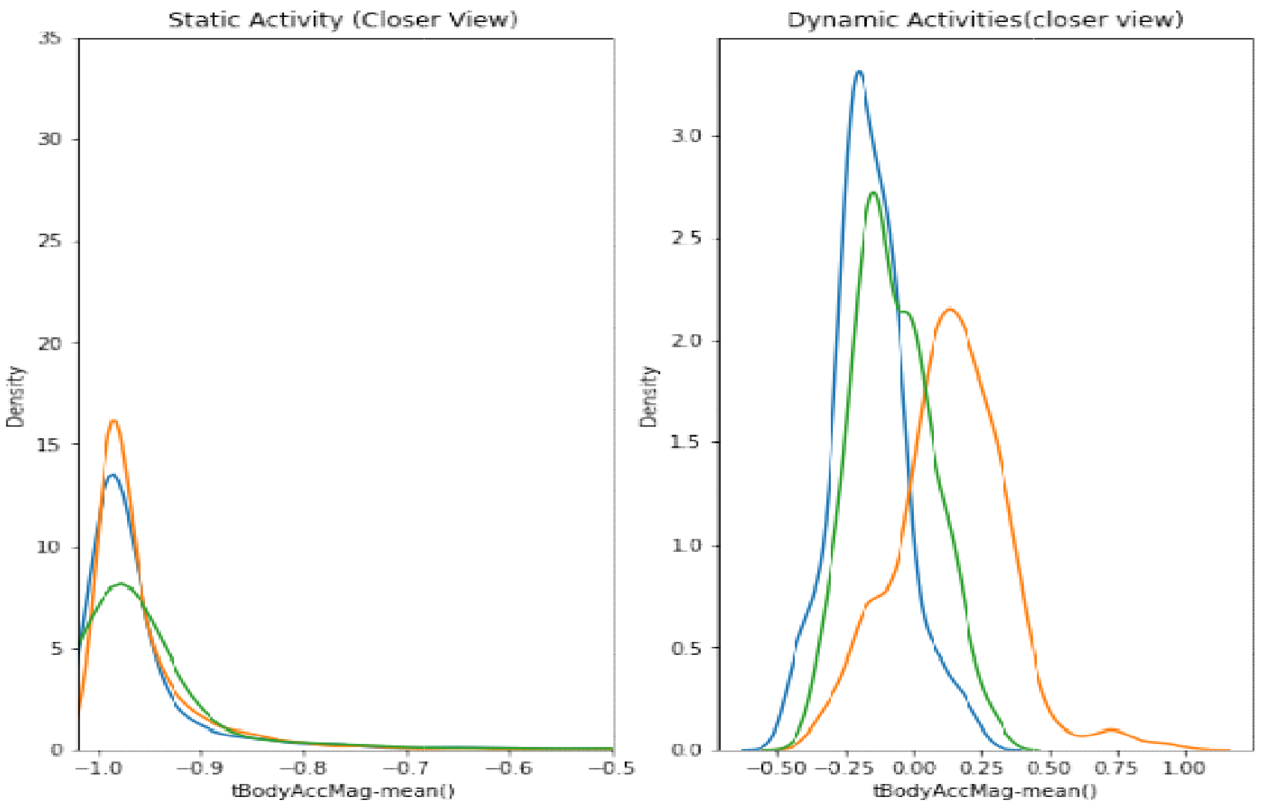
```
In [4]:  print('Number of duplicates in train : ',sum(train. duplicated()))
          print('Number of duplicates in test : ', sum(test. duplicated()))

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

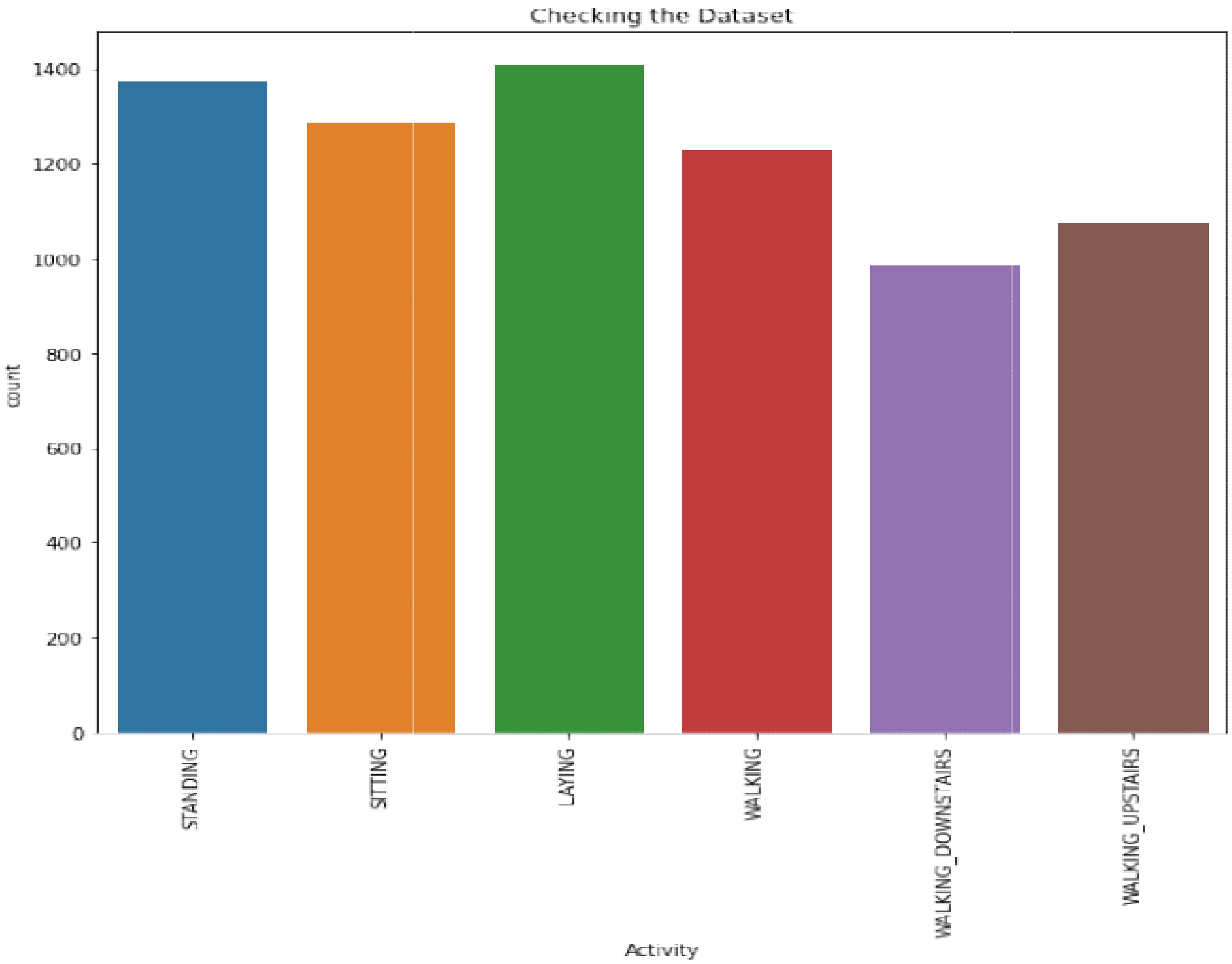


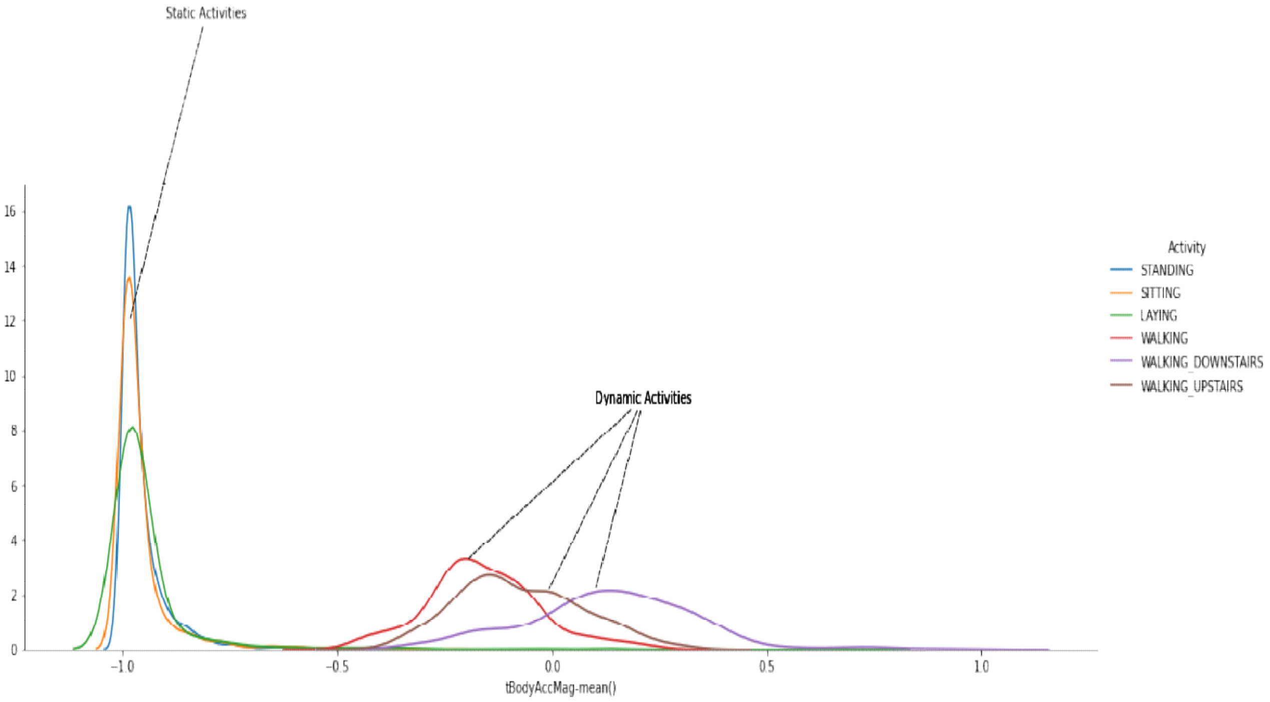
```
In [5]:  print('Total number of null values in train:',train. isna(). values. sum())
          print('Total number of null values in test:',test. isna(). values. sum())

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

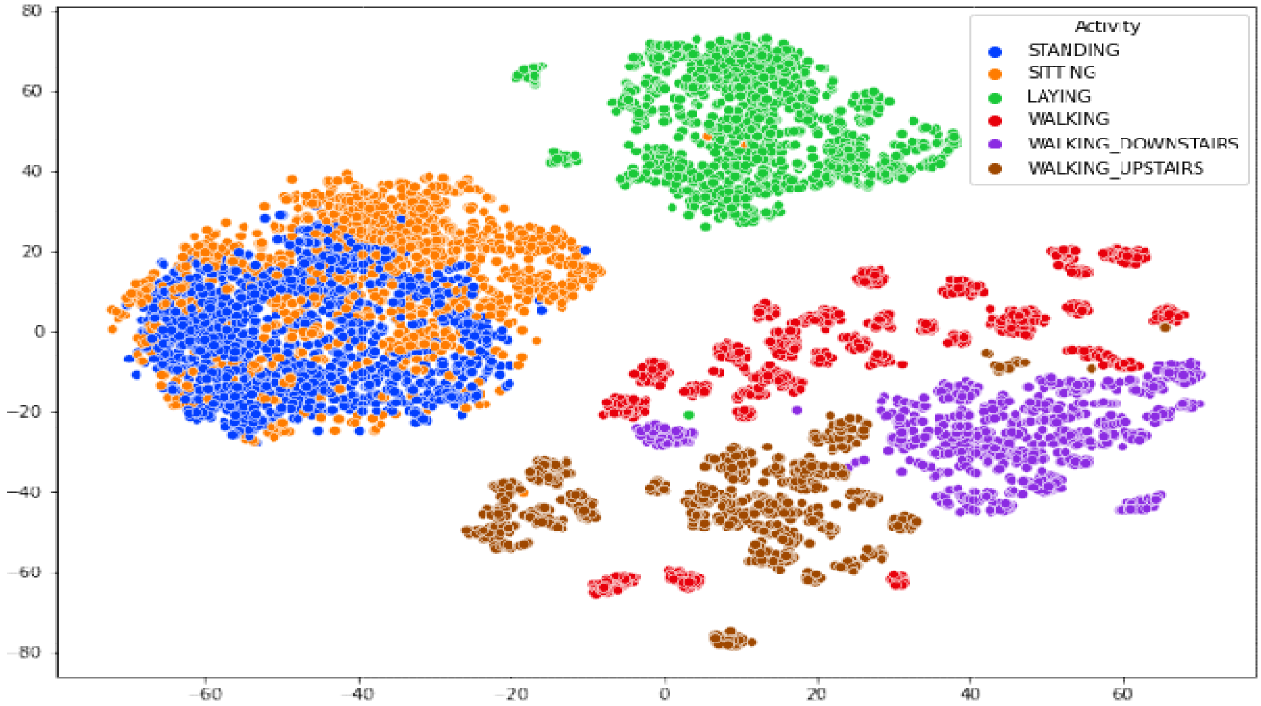


```
In [11]: # Checking wether the classs are imbalanced or not
         plt.figure(figsize =(10,8))
         sns.countplot(train['Activity'])
         plt.title('Checking the Dataset')
         plt.xticks(rotation = 90)
         plt.show()
```

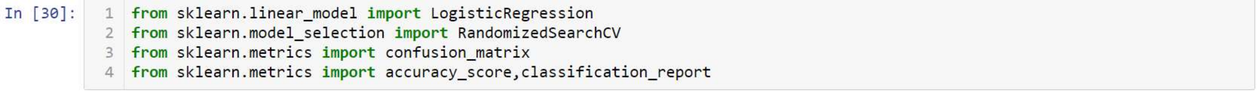



Checking the Dataset
1400
1200
1000
800
600
400
200
0
count
STANDING
SITTING
LAYING
WALKING
WALKING_DOWNSTAIRS
WALKING_UPSTAIRS
Activity


Static Activities
Dynamic Activities
16
14
12
10
8
6
4
2
0
-1.0
-0.5
0.0
0.5
1.0
tBodyAccMag-mean()
Activity
STANDING
SITTING
LAYING
WALKING
WALKING_DOWNSTAIRS
WALKING_UPSTAIRS

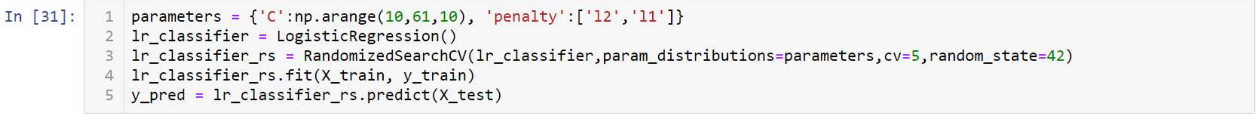


3. Implementing Various Classification Algorithm



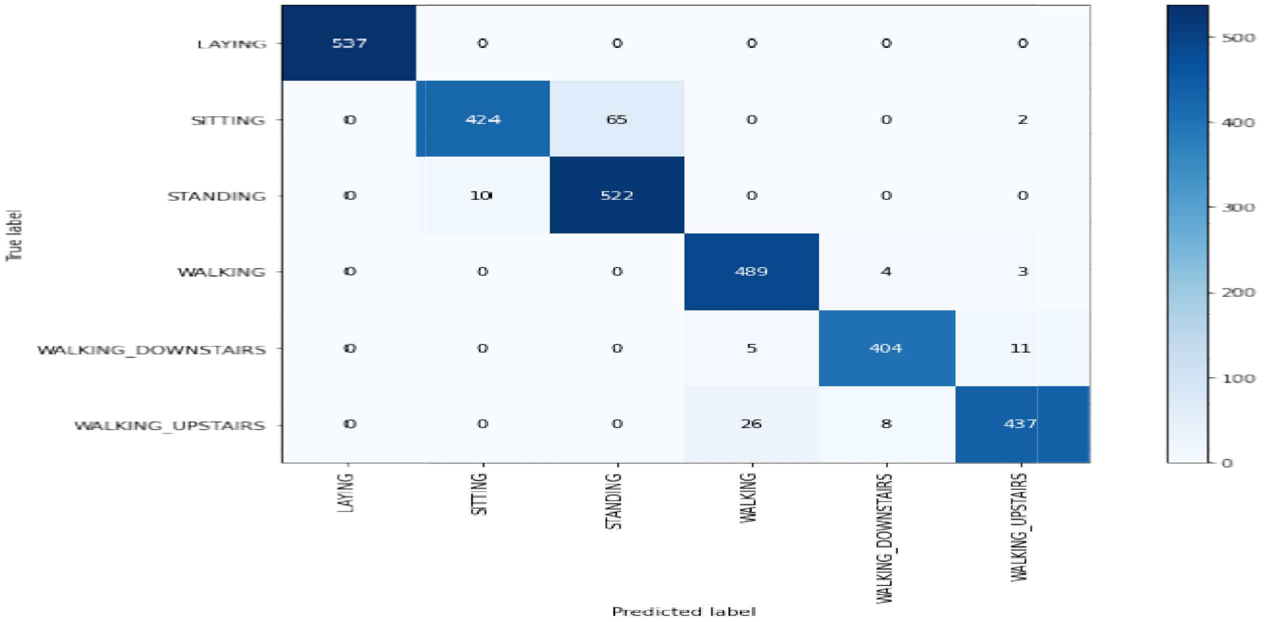
In [30]:

```
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
```

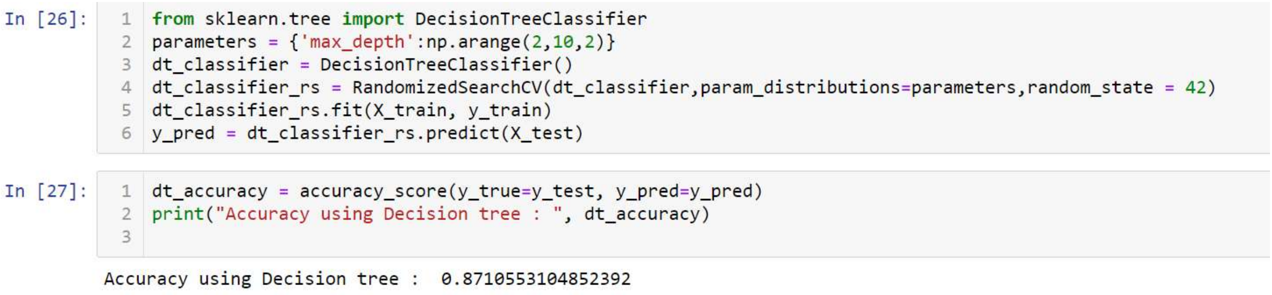


In [31]:

```
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)
```



Decision Tree Classifier

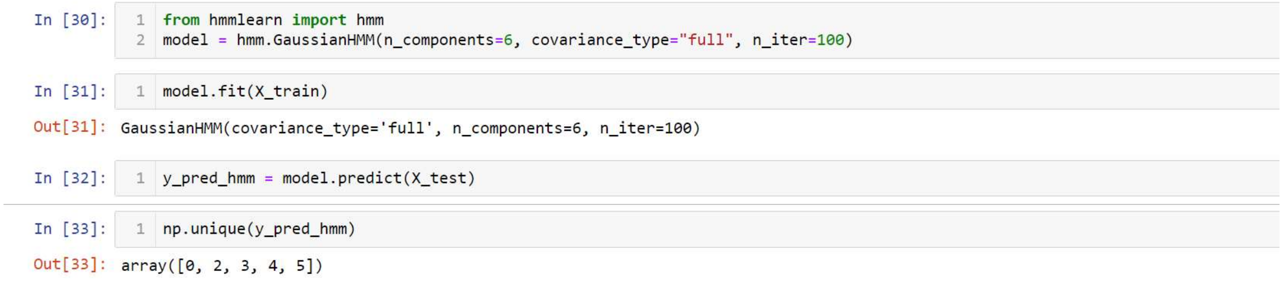


```
In [26]: from sklearn.tree import DecisionTreeClassifier
         parameters = {'max_depth':np.arange(2,10,2)}
         dt_classifier = DecisionTreeClassifier()
         dt_classifier_rs = RandomizedSearchCV(dt_classifier,param_distributions=parameters,random_state = 42)
         dt_classifier_rs.fit(X_train, y_train)
         y_pred = dt_classifier_rs.predict(X_test)
```

```
In [27]: dt_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
         print("Accuracy using Decision tree : ", dt_accuracy)

```

```
Accuracy using Decision tree :  0.8710553104852392
```

HMM (Hidden Markov Model)



```
In [30]: from hmmlearn import hmm
         model = hmm.GaussianHMM(n_components=6, covariance_type="full", n_iter=100)
```

```
In [31]: model.fit(X_train)
```

```
Out[31]: GaussianHMM(covariance_type='full', n_components=6, n_iter=100)
```

```
In [32]: y_pred_hmm = model.predict(X_test)
```

```
In [33]: np.unique(y_pred_hmm)
```

```
Out[33]: array([0, 2, 3, 4, 5])
```