SURUTHI S ¶

225229141

In [1]:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')


```

Step-1: [Understand Data]

In [2]:

```
df=pd.read_csv("Human_Activity_Data.csv")
df.head()
```

Out[2]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAc mad()-
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.92352
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.95768
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.97746
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.98930
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.99044

5 rows × 562 columns

In [3]:

```
df.shape
```

Out[3]:

```
(10299, 562)
```

In [4]:

```
df.size
```

Out[4]:

```
5788038
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10299 entries, 0 to 10298
Columns: 562 entries, tBodyAcc-mean()-X to Activity
dtypes: float64(561), object(1)
memory usage: 44.2+ MB
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['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',
       ...
       'fBodyBodyGyroJerkMag-skewness()', 'fBodyBodyGyroJerkMag-kurtosis()',
       'angle(tBodyAccMean,gravity)', 'angle(tBodyAccJerkMean),gravityMean)',
       'angle(tBodyGyroMean,gravityMean)',
       'angle(tBodyGyroJerkMean,gravityMean)', 'angle(X,gravityMean)',
       'angle(Y,gravityMean)', 'angle(Z,gravityMean)', 'Activity'],
      dtype='object', length=562)
```

In [7]:

```
df['angle(Z,gravityMean)'].value_counts
```

Out[7]:

```
<bound method IndexOpsMixin.value_counts of 0       -0.058627
1       -0.054317
2       -0.049118
3       -0.047663
4       -0.043892
           ...   
10294    0.184784
10295    0.182412
10296    0.181184
10297    0.187563
10298    0.188103
Name: angle(Z,gravityMean), Length: 10299, dtype: float64>
```

Step-2: [Build a small Dataset

In [8]:

```
import numpy as np
a=df[df['Activity']=='LAYING'].head(500)
b=df[df['Activity']=='SITTING'].head(500)
c=df[df['Activity']=='WALKING'].head(500)
```

In [9]:

```
newdf=pd.concat([a,b,c])
```

In [10]:

```
newdf.shape
```

Out[10]:

```
(1500, 562)
```

In [11]:

```
newdf.to_csv("Human_Activity_new.csv")
```

In [1]:

```
from sklearn.metrics import classification_report

from sklearn.ensemble import GradientBoostingClassifier,AdaBoostClassifier

from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LogisticRegressionCV

from sklearn.ensemble import RandomForestClassifier, VotingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import cross_val_score
```

Step-3: [Build GradientBoostingClassifier]

In [12]:

```
df = pd.read_csv("Human_Activity_new.csv")
```

In [13]:

```
df.head()
```

Out[13]:

	Unnamed: 0	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc mad()-
0	51	0.403474	-0.015074	-0.118167	-0.914811	-0.895231	-0.891748	-0.917696	-0.92462
1	52	0.278373	-0.020561	-0.096825	-0.984883	-0.991118	-0.982112	-0.987985	-0.99036
2	53	0.276555	-0.017869	-0.107621	-0.994195	-0.996372	-0.995615	-0.994901	-0.99636
3	54	0.279575	-0.017276	-0.109481	-0.996135	-0.995812	-0.998689	-0.996393	-0.99547
4	55	0.276527	-0.016819	-0.107983	-0.996775	-0.997256	-0.995422	-0.997167	-0.99710

5 rows × 563 columns

In [14]:

```
df.shape
```

Out[14]:

```
(1500, 563)
```

In [15]:

```
df.size
```

Out[15]:

```
844500
```

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Columns: 563 entries, Unnamed: 0 to Activity
dtypes: float64(561), int64(1), object(1)
memory usage: 6.4+ MB
```

In [17]:

```
df.columns
```

Out[17]:

```
Index(['Unnamed: 0', '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',
       ...
       'fBodyBodyGyroJerkMag-skewness()', 'fBodyBodyGyroJerkMag-kurtosis()',
       'angle(tBodyAccMean,gravity)', 'angle(tBodyAccJerkMean),gravityMean)',
       'angle(tBodyGyroMean,gravityMean)',
       'angle(tBodyGyroJerkMean,gravityMean)', 'angle(X,gravityMean)',
       'angle(Y,gravityMean)', 'angle(Z,gravityMean)', 'Activity'],
      dtype='object', length=563)
```

In [18]:
```
df.describe()
```

Out[18]:

	Unnamed: 0	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBc
count	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500
mean	1430.972000	0.270425	-0.015542	-0.108074	-0.751373	-0.597033	-0.706049	-0
std	845.331241	0.084685	0.036471	0.055224	0.317106	0.490449	0.367092	0
min	27.000000	-1.000000	-0.684097	-1.000000	-0.999300	-0.998524	-0.998689	-0
25%	726.750000	0.264859	-0.021433	-0.118534	-0.993145	-0.983467	-0.982370	-0
50%	1407.500000	0.276946	-0.016817	-0.108755	-0.966535	-0.937492	-0.940266	-0
75%	2133.250000	0.285803	-0.011554	-0.100423	-0.392574	-0.057412	-0.438577	-0
max	3102.000000	0.559135	0.324130	0.543939	0.057201	0.671192	0.458721	0

8 rows × 562 columns

In [19]:
```
X = df.drop('Activity',axis=1)
X.head()
```

Out[19]:

	Unnamed: 0	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc mad()-
0	51	0.403474	-0.015074	-0.118167	-0.914811	-0.895231	-0.891748	-0.917696	-0.92462
1	52	0.278373	-0.020561	-0.096825	-0.984883	-0.991118	-0.982112	-0.987985	-0.99036
2	53	0.276555	-0.017869	-0.107621	-0.994195	-0.996372	-0.995615	-0.994901	-0.99636
3	54	0.279575	-0.017276	-0.109481	-0.996135	-0.995812	-0.998689	-0.996393	-0.99547
4	55	0.276527	-0.016819	-0.107983	-0.996775	-0.997256	-0.995422	-0.997167	-0.99710

5 rows × 562 columns

In [20]:
```
y=df['Activity']
y.head()
```

Out[20]:
```
0    LAYING
1    LAYING
2    LAYING
3    LAYING
4    LAYING
Name: Activity, dtype: object
```

In [21]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=
```

In [22]:
```
model = GradientBoostingClassifier(subsample=0.5,n_estimators=100,learning_rate=
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

In [23]:
```
print(accuracy_score(y_test,y_pred))
```

```
1.0
```

```
In [24]: print(classification_report(y_test,y_pred))

```

```
              precision    recall  f1-score   support

      LAYING       1.00      1.00      1.00       148
     SITTING       1.00      1.00      1.00       141
     WALKING       1.00      1.00      1.00       161

    accuracy                           1.00       450
   macro avg       1.00      1.00      1.00       450
weighted avg       1.00      1.00      1.00       450
```

Step-4: [Find Best no. of trees and Best Learning Rate using Grid Search and Cross Validation]

```
In [25]: all_scores = cross_val_score(estimator=model, X=X_train, y=y_train, cv=5)
```

```
In [26]: print(all_scores)
```

```
[1.         1.         1.         1.         0.9952381]
```

```
In [27]: all_scores.mean()
```

```
Out[27]: 0.9990476190476191
```

```
In [28]: parameter = {'n_estimators': [50, 100, 200, 400], 'learning_rate': [0.1, 0.01]}
```

```
In [29]: model1 = GridSearchCV(estimator=model,
         param_grid=parameter,cv=5,n_jobs=-1)
```

```
In [30]: model1.fit(X_train,y_train)
```

```
Out[30]: GridSearchCV(cv=5,
                      estimator=GradientBoostingClassifier(learning_rate=1.0,
                                                           max_depth=10, subsample=0.5),
                      n_jobs=-1,
                      param_grid={'learning_rate': [0.1, 0.01],
                                  'n_estimators': [50, 100, 200, 400]})
```

```
In [31]: y_pred2=model1.predict(X_test)
```

```
In [33]: print(classification_report(y_test,y_pred2))
```

```
              precision    recall  f1-score   support

      LAYING       1.00      1.00      1.00       148
     SITTING       1.00      1.00      1.00       141
     WALKING       1.00      1.00      1.00       161

    accuracy                           1.00       450
   macro avg       1.00      1.00      1.00       450
weighted avg       1.00      1.00      1.00       450
```

In [34]:
```python
print(model1.best_estimator_)
```

```
GradientBoostingClassifier(max_depth=10, n_estimators=50, subsample=0.5)
```

step 5: [Best AdaBoostClassifier]

In [35]:
```python
base = DecisionTreeClassifier(max_features=4)
model2 = AdaBoostClassifier(base_estimator=base,random_state=0)
param_grid = {'n_estimators': [100, 150, 200], 'learning_rate': [0.01, 0.001]}
model3 = GridSearchCV(model2,param_grid,cv=5,n_jobs=-1)
model3.fit(X_train,y_train)
```

Out[35]:
```
GridSearchCV(cv=5,
             estimator=AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max
_features=4),
                                          random_state=0),
             n_jobs=-1,
             param_grid={'learning_rate': [0.01, 0.001],
                         'n_estimators': [100, 150, 200]})
```

In [36]:
```python
y_pred3=model3.predict(X_test)
y_pred3
```

Out[36]:
```
array(['WALKING', 'WALKING', 'LAYING', 'LAYING', 'SITTING', 'SITTING',
       'WALKING', 'SITTING', 'WALKING', 'LAYING', 'LAYING', 'WALKING',
       'SITTING', 'SITTING', 'LAYING', 'WALKING', 'WALKING', 'SITTING',
       'LAYING', 'LAYING', 'WALKING', 'WALKING', 'WALKING', 'WALKING',
       'WALKING', 'LAYING', 'WALKING', 'WALKING', 'WALKING', 'WALKING',
       'WALKING', 'LAYING', 'WALKING', 'WALKING', 'LAYING', 'SITTING',
       'LAYING', 'LAYING', 'WALKING', 'SITTING', 'WALKING', 'SITTING',
       'WALKING', 'SITTING', 'WALKING', 'LAYING', 'LAYING', 'LAYING',
       'LAYING', 'SITTING', 'LAYING', 'SITTING', 'WALKING', 'WALKING',
       'SITTING', 'LAYING', 'SITTING', 'LAYING', 'SITTING', 'SITTING',
       'SITTING', 'SITTING', 'LAYING', 'LAYING', 'SITTING', 'SITTING',
       'LAYING', 'WALKING', 'WALKING', 'LAYING', 'LAYING', 'SITTING',
       'SITTING', 'LAYING', 'SITTING', 'WALKING', 'SITTING', 'SITTING',
       'SITTING', 'LAYING', 'WALKING', 'LAYING', 'LAYING', 'SITTING',
       'LAYING', 'LAYING', 'SITTING', 'SITTING', 'SITTING', 'LAYING',
       'WALKING', 'LAYING', 'LAYING', 'WALKING', 'WALKING', 'SITTING',
       'LAYING', 'LAYING', 'WALKING', 'LAYING', 'SITTING', 'SITTING',
       'WALKING', 'WALKING', 'LAYING', 'LAYING', 'LAYING', 'WALKING',
       'LAYING', 'SITTING', 'LAYING', 'WALKING', 'SITTING', 'LAYING',
       'LAYING', 'LAYING', 'SITTING', 'LAYING', 'LAYING', 'WALKING',
```

In [38]:
```python
print(classification_report(y_test,y_pred3))
```

```
              precision    recall  f1-score   support

      LAYING       0.84      0.86      0.85       148
     SITTING       0.85      0.84      0.84       141
     WALKING       1.00      0.99      0.99       161

    accuracy                           0.90       450
   macro avg       0.90      0.90      0.90       450
weighted avg       0.90      0.90      0.90       450
```

In [39]:
```python
print(model3.best_estimator_)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_features=4),
                   learning_rate=0.01, n_estimators=100, random_state=0)
```

Step-6: [Build a LogisticRegressionCV classifier]

In [40]:
```
model4 = LogisticRegressionCV(cv=4,Cs=5,penalty='l2')
model4.fit(X_train,y_train)
y_pred2=model4.predict(X_test)
y_pred2
```

```
       'WALKING', 'WALKING', 'WALKING', 'LAYING', 'SITTING', 'WALKING',
       'WALKING', 'WALKING', 'WALKING', 'WALKING', 'WALKING', 'WALKING',
       'SITTING', 'WALKING', 'SITTING', 'LAYING', 'SITTING', 'SITTING',
       'WALKING', 'WALKING', 'WALKING', 'SITTING', 'WALKING', 'LAYING',
       'WALKING', 'SITTING', 'WALKING', 'SITTING', 'LAYING', 'SITTING',
       'SITTING', 'WALKING', 'WALKING', 'LAYING', 'SITTING', 'SITTING',
       'WALKING', 'WALKING', 'SITTING', 'LAYING', 'WALKING', 'SITTING',
       'LAYING', 'LAYING', 'LAYING', 'WALKING', 'SITTING', 'WALKING',
       'WALKING', 'WALKING', 'LAYING', 'WALKING', 'WALKING', 'WALKING',
       'WALKING', 'SITTING', 'LAYING', 'SITTING', 'LAYING', 'WALKING',
       'LAYING', 'WALKING', 'SITTING', 'WALKING', 'LAYING', 'WALKING',
       'SITTING', 'SITTING', 'SITTING', 'WALKING', 'WALKING', 'SITTING',
       'LAYING', 'SITTING', 'WALKING', 'WALKING', 'LAYING', 'LAYING',
       'WALKING', 'WALKING', 'SITTING', 'LAYING', 'LAYING', 'WALKING',
       'LAYING', 'SITTING', 'WALKING', 'SITTING', 'SITTING', 'SITTING',
       'LAYING', 'WALKING', 'SITTING', 'WALKING', 'SITTING', 'SITTING',
       'SITTING', 'WALKING', 'LAYING', 'WALKING', 'LAYING', 'LAYING',
       'WALKING', 'SITTING', 'WALKING', 'SITTING', 'SITTING', 'WALKING',
       'LAYING', 'LAYING', 'SITTING', 'SITTING', 'LAYING', 'SITTING',
       'WALKING', 'WALKING', 'WALKING', 'SITTING', 'LAYING', 'SITTING',
```

In [42]:
```
print(classification_report(y_test,y_pred2))
```

```
              precision    recall  f1-score   support

      LAYING       1.00      1.00      1.00       148
     SITTING       1.00      0.99      1.00       141
     WALKING       0.99      1.00      1.00       161

    accuracy                           1.00       450
   macro avg       1.00      1.00      1.00       450
weighted avg       1.00      1.00      1.00       450
```

Step-7: [Build VotingClassifier]

In [43]:
```
model4=VotingClassifier(estimators=[('lr',model4),('gbc',model1)], voting='hard'
model4.fit(X_train,y_train)
```

Out[43]:
```
VotingClassifier(estimators=[('lr', LogisticRegressionCV(Cs=5, cv=4)),
                             ('gbc',
                              GridSearchCV(cv=5,
                                           estimator=GradientBoostingClassifier(lea
rning_rate=1.0,
                                                                                max
_depth=10,
                                                                                sub
sample=0.5),
                                           n_jobs=-1,
                                           param_grid={'learning_rate': [0.1,
                                                                         0.01],
                                                       'n_estimators': [50, 100,
                                                                        200,
                                                                        400]}))])
```

In [44]:

```
y_pred3=model4.predict(X_test)
y_pred3
```

```
       'WALKING', 'WALKING', 'WALKING', 'LAYING', 'SITTING', 'WALKING',
       'WALKING', 'WALKING', 'WALKING', 'WALKING', 'WALKING', 'WALKING',
       'SITTING', 'WALKING', 'SITTING', 'LAYING', 'SITTING', 'SITTING',
       'WALKING', 'WALKING', 'WALKING', 'SITTING', 'WALKING', 'LAYING',
       'WALKING', 'SITTING', 'WALKING', 'SITTING', 'LAYING', 'SITTING',
       'SITTING', 'WALKING', 'WALKING', 'LAYING', 'SITTING', 'SITTING',
       'WALKING', 'WALKING', 'SITTING', 'LAYING', 'WALKING', 'SITTING',
       'LAYING', 'LAYING', 'LAYING', 'WALKING', 'SITTING', 'WALKING',
       'WALKING', 'WALKING', 'LAYING', 'WALKING', 'WALKING', 'WALKING',
       'WALKING', 'SITTING', 'LAYING', 'SITTING', 'LAYING', 'WALKING',
       'LAYING', 'WALKING', 'SITTING', 'WALKING', 'LAYING', 'WALKING',
       'SITTING', 'SITTING', 'SITTING', 'WALKING', 'WALKING', 'SITTING',
       'LAYING', 'SITTING', 'WALKING', 'WALKING', 'LAYING', 'LAYING',
       'WALKING', 'WALKING', 'SITTING', 'LAYING', 'LAYING', 'WALKING',
       'LAYING', 'SITTING', 'WALKING', 'SITTING', 'SITTING', 'SITTING',
       'LAYING', 'WALKING', 'SITTING', 'WALKING', 'SITTING', 'SITTING',
       'SITTING', 'WALKING', 'LAYING', 'WALKING', 'LAYING', 'LAYING',
       'WALKING', 'SITTING', 'WALKING', 'SITTING', 'SITTING', 'WALKING',
       'LAYING', 'LAYING', 'SITTING', 'SITTING', 'LAYING', 'SITTING',
       'WALKING', 'WALKING', 'WALKING', 'SITTING', 'LAYING', 'SITTING',
```

In [46]:

```
print(classification_report(y_test,y_pred3))
```

```
              precision    recall  f1-score   support

      LAYING       1.00      1.00      1.00       148
     SITTING       1.00      1.00      1.00       141
     WALKING       1.00      1.00      1.00       161

    accuracy                           1.00       450
   macro avg       1.00      1.00      1.00       450
weighted avg       1.00      1.00      1.00       450
```

Step-8: [Interpret your results]

In [47]:

```
print(model1.best_estimator_)
```

```
GradientBoostingClassifier(max_depth=10, n_estimators=50, subsample=0.5)
```

In [48]:

```
print(model3.best_estimator_)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_features=4),
                   learning_rate=0.01, n_estimators=100, random_state=0)
```

GradientBoostingClassifier

```
In [49]:
classifierF = GradientBoostingClassifier(n_estimators=50,max_features=4)
all_scoresF = cross_val_score(estimator=classifierF, X=X_train, y=y_train, cv=5)
parameter = {'n_estimators': [50, 100, 200, 400], 'learning_rate': [0.1, 0.01]}
modelGB = GridSearchCV(estimator=classifierF,param_grid=parameter,cv=5,n_jobs=-1
modelGB.fit(X_train,y_train)
```

```
Out[49]: GridSearchCV(cv=5,
                      estimator=GradientBoostingClassifier(max_features=4,
                                                           n_estimators=50),
                      n_jobs=-1,
                      param_grid={'learning_rate': [0.1, 0.01],
                                  'n_estimators': [50, 100, 200, 400]})
```

```
In [50]:
y_predGB=model3.predict(X_test)
y_predGB
```

```
       'LAYING', 'LAYING', 'SITTING', 'SITTING', 'SITTING', 'SITTING',
       'WALKING', 'WALKING', 'WALKING', 'SITTING', 'LAYING', 'SITTING',
       'LAYING', 'WALKING', 'LAYING', 'SITTING', 'LAYING', 'WALKING',
       'SITTING', 'SITTING', 'SITTING', 'SITTING', 'SITTING', 'LAYING',
       'SITTING', 'SITTING', 'WALKING', 'WALKING', 'WALKING', 'SITTING',
       'LAYING', 'SITTING', 'LAYING', 'WALKING', 'LAYING', 'WALKING',
       'WALKING', 'SITTING', 'WALKING', 'SITTING', 'WALKING', 'WALKING',
       'WALKING', 'LAYING', 'WALKING', 'WALKING', 'WALKING', 'SITTING',
       'SITTING', 'LAYING', 'WALKING', 'WALKING', 'LAYING', 'WALKING',
       'SITTING', 'LAYING', 'WALKING', 'LAYING', 'WALKING', 'SITTING',
       'SITTING', 'WALKING', 'WALKING', 'SITTING', 'SITTING', 'LAYING',
       'WALKING', 'WALKING', 'LAYING', 'LAYING', 'LAYING', 'LAYING',
       'LAYING', 'WALKING', 'WALKING', 'LAYING', 'SITTING', 'LAYING',
       'SITTING', 'SITTING', 'LAYING', 'WALKING', 'SITTING', 'WALKING',
       'LAYING', 'LAYING', 'LAYING', 'SITTING', 'LAYING', 'WALKING',
       'SITTING', 'LAYING', 'LAYING', 'WALKING', 'SITTING', 'SITTING',
       'SITTING', 'LAYING', 'SITTING', 'SITTING', 'LAYING', 'LAYING',
       'LAYING', 'SITTING', 'WALKING', 'SITTING', 'LAYING', 'WALKING',
       'LAYING', 'WALKING', 'SITTING', 'SITTING', 'WALKING', 'WALKING'],
      dtype=object)
```

```
In [52]:
print(classification_report(y_test,y_predGB))
```

```
              precision    recall  f1-score   support

      LAYING       0.84      0.86      0.85       148
     SITTING       0.85      0.84      0.84       141
     WALKING       1.00      0.99      0.99       161

    accuracy                           0.90       450
   macro avg       0.90      0.90      0.90       450
weighted avg       0.90      0.90      0.90       450
```

AdaBoostClassifier

In [53]:
```
modelABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
learning_rate=0.01,
n_estimators=100,
random_state=0)
param_grid = {'n_estimators': [100, 150, 200], 'learning_rate': [0.01, 0.001]}
modelGSCV = GridSearchCV(modelABC,param_grid,cv=5,n_jobs=-1)
modelGSCV.fit(X_train,y_train)
```

Out[53]:
```
GridSearchCV(cv=5,
             estimator=AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
                                          learning_rate=0.01, n_estimators=100,
                                          random_state=0),
             n_jobs=-1,
             param_grid={'learning_rate': [0.01, 0.001],
                         'n_estimators': [100, 150, 200]})
```

In [54]:
```
y_predGSCV=model3.predict(X_test)
y_predGSCV
```

```
       'LAYING', 'LAYING', 'LAYING', 'WALKING', 'SITTING', 'WALKING',
       'WALKING', 'WALKING', 'LAYING', 'WALKING', 'WALKING', 'WALKING',
       'WALKING', 'SITTING', 'LAYING', 'SITTING', 'LAYING', 'WALKING',
       'SITTING', 'WALKING', 'SITTING', 'WALKING', 'LAYING', 'WALKING',
       'SITTING', 'SITTING', 'LAYING', 'WALKING', 'WALKING', 'LAYING',
       'LAYING', 'SITTING', 'WALKING', 'WALKING', 'LAYING', 'LAYING',
       'WALKING', 'WALKING', 'LAYING', 'LAYING', 'LAYING', 'WALKING',
       'LAYING', 'SITTING', 'LAYING', 'SITTING', 'SITTING', 'SITTING',
       'LAYING', 'WALKING', 'SITTING', 'WALKING', 'SITTING', 'LAYING',
       'SITTING', 'WALKING', 'LAYING', 'WALKING', 'LAYING', 'LAYING',
       'WALKING', 'SITTING', 'WALKING', 'SITTING', 'SITTING', 'WALKING',
       'LAYING', 'LAYING', 'SITTING', 'SITTING', 'SITTING', 'SITTING',
       'WALKING', 'WALKING', 'WALKING', 'SITTING', 'LAYING', 'SITTING',
       'LAYING', 'WALKING', 'LAYING', 'SITTING', 'LAYING', 'WALKING',
       'SITTING', 'SITTING', 'SITTING', 'SITTING', 'SITTING', 'LAYING',
       'SITTING', 'SITTING', 'WALKING', 'WALKING', 'WALKING', 'SITTING',
       'LAYING', 'SITTING', 'LAYING', 'WALKING', 'LAYING', 'WALKING',
       'WALKING', 'SITTING', 'WALKING', 'SITTING', 'WALKING', 'WALKING',
       'WALKING', 'LAYING', 'WALKING', 'WALKING', 'WALKING', 'SITTING',
       'SITTING', 'LAYING', 'WALKING', 'WALKING', 'LAYING', 'WALKING',
```

In [56]:
```
print(classification_report(y_test,y_predGSCV))
```

```
              precision    recall  f1-score   support

      LAYING       0.84      0.86      0.85       148
     SITTING       0.85      0.84      0.84       141
     WALKING       1.00      0.99      0.99       161

    accuracy                           0.90       450
   macro avg       0.90      0.90      0.90       450
weighted avg       0.90      0.90      0.90       450
```