Lab 10: Patients Physical Activities prediction using Boosting

Name : Murali kumar R

Roll no : 225229120

Step 1: Understand Data

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import precision_score, recall_score,accuracy_score,roc_a
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
```

In [2]:
```
df = pd.read_csv("Human_Activity_Data.csv")
```

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

Out[3]:

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

5 rows × 562 columns

In [4]:
```
df.shape
```

Out[4]:
```
(10299, 562)
```

In [5]:
```
df.columns
```

Out[5]:
```
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 [6]:
```
df.dtypes
```

Out[6]:
```
tBodyAcc-mean()-X                       float64
tBodyAcc-mean()-Y                       float64
tBodyAcc-mean()-Z                       float64
tBodyAcc-std()-X                        float64
tBodyAcc-std()-Y                        float64
                                          ...
angle(tBodyGyroJerkMean,gravityMean)    float64
angle(X,gravityMean)                    float64
angle(Y,gravityMean)                    float64
angle(Z,gravityMean)                    float64
Activity                                 object
Length: 562, dtype: object
```

In [7]:
```
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 [8]:
```
df["Activity"].value_counts()
```

Out[8]:
```
LAYING                1944
STANDING              1906
SITTING               1777
WALKING               1722
WALKING_UPSTAIRS      1544
WALKING_DOWNSTAIRS    1406
Name: Activity, dtype: int64
```

Step 2: Build a small dataset

In [9]:
```
lay = df.loc[df['Activity'] == "LAYING"][:500]
sit = df.loc[df['Activity'] == "SITTING"][:500]
walk = df.loc[df['Activity'] == "WALKING"][:500]
frames = [lay, sit, walk]
df_new = pd.concat(frames)
```

In [10]:
```
df_new.shape
```

Out[10]: (1500, 562)

Store this dataframe as a new csv file

In [11]:
```
df_new.to_csv("Human_Activity_sample.csv")
```

Step 3: Build GradientBoostingClassifier

Import your reduced csv file

In [12]:
```
df1= pd.read_csv('Human_Activity_sample.csv')
```

Print Basic Properties of new csv file

In [13]:
```
df1.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	t
0	51	0.403474	-0.015074	-0.118167	-0.914811	-0.895231	-0.891748	-0.917696	
1	52	0.278373	-0.020561	-0.096825	-0.984883	-0.991118	-0.982112	-0.987985	
2	53	0.276555	-0.017869	-0.107621	-0.994195	-0.996372	-0.995615	-0.994901	
3	54	0.279575	-0.017276	-0.109481	-0.996135	-0.995812	-0.998689	-0.996393	
4	55	0.276527	-0.016819	-0.107983	-0.996775	-0.997256	-0.995422	-0.997167	

5 rows × 563 columns

In [14]:
```
df1.shape
```

Out[14]: (1500, 563)

In [15]:
```
df1.columns
```

Out[15]:
```
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 [16]:
```
df1.dtypes
```

Out[16]:
```
Unnamed: 0                              int64
tBodyAcc-mean()-X                     float64
tBodyAcc-mean()-Y                     float64
tBodyAcc-mean()-Z                     float64
tBodyAcc-std()-X                      float64
                                        ...
angle(tBodyGyroJerkMean,gravityMean)  float64
angle(X,gravityMean)                  float64
angle(Y,gravityMean)                  float64
angle(Z,gravityMean)                  float64
Activity                               object
Length: 563, dtype: object
```

In [17]:
```
df1.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 [18]:
```
df1["Activity"].value_counts()
```

Out[18]:
```
LAYING     500
SITTING    500
WALKING    500
Name: Activity, dtype: int64
```

In [19]:
```
X=df1.drop('Activity',axis=1)
X
```

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
0	51	0.403474	-0.015074	-0.118167	-0.914811	-0.895231	-0.891748	-0.91769
1	52	0.278373	-0.020561	-0.096825	-0.984883	-0.991118	-0.982112	-0.98798
2	53	0.276555	-0.017869	-0.107621	-0.994195	-0.996372	-0.995615	-0.99490
3	54	0.279575	-0.017276	-0.109481	-0.996135	-0.995812	-0.998689	-0.99639
4	55	0.276527	-0.016819	-0.107983	-0.996775	-0.997256	-0.995422	-0.99716
...	...	...	...	...	...	...	...	...
1495	2637	0.224104	-0.070152	-0.160533	-0.345152	0.143162	-0.430728	-0.38157
1496	2638	0.218667	-0.000151	-0.129555	-0.358372	0.044663	-0.489753	-0.39883
1497	2639	0.282650	0.010504	-0.105785	-0.430536	-0.031534	-0.515552	-0.44829
1498	2640	0.310453	-0.005621	-0.116935	-0.439045	-0.092861	-0.498774	-0.46758
1499	2641	0.350868	-0.029195	-0.120600	-0.367631	0.025890	-0.428028	-0.41591

1500 rows × 562 columns

In [20]:
```
y=df1.Activity
y
```

Out[20]:
```
0         LAYING
1         LAYING
2         LAYING
3         LAYING
4         LAYING
          ...
1495    WALKING
1496    WALKING
1497    WALKING
1498    WALKING
1499    WALKING
Name: Activity, Length: 1500, dtype: object
```

Split into training and testing set

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

Create GradientBoostingClasssifier,fit and predict

In [22]:
```
model = GradientBoostingClassifier(n_estimators=100,learning_rate=1.0,max_dept
model.fit(X_train,y_train)
```

Out[22]:
```
GradientBoostingClassifier(learning_rate=1.0, max_depth=1, random_state=42)
```

In [23]:
```
y_pred=model.predict(X_test)
```

Print accuracy and classification report

In [24]:
```
accuracy_score(y_test,y_pred)
```

Out[24]:
```
1.0
```

In [25]:
```
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
```

Step4. [Find Best no. of trees and Best Learning Rate using Grid Search and Cross Validation]

In [26]:
```
classifier = GradientBoostingClassifier()
```

In [27]:
```
all_scores = cross_val_score(estimator=classifier, X=X_train, y=y_train, cv=5)
```

In [28]:
```
all_scores
```

Out[28]:
```
array([1., 1., 1., 1., 1.])
```

To find the average of all the accuracies, simple use the mean() method

In [29]:
```
all_scores.mean()
```

Out[29]:
```
1.0
```

In [30]:
```
parameter = {'n_estimators': [50, 100, 200, 400], 'learning_rate': [0.1, 0.01]
```

In [31]:
```
model1 = GridSearchCV(estimator=classifier, param_grid=parameter,cv=5, n_jobs=
```

In [32]:
```
model1.fit(X_train,y_train)
```

Out[32]:
```
GridSearchCV(cv=5, estimator=GradientBoostingClassifier(), n_jobs=-1,
             param_grid={'learning_rate': [0.1, 0.01],
                         'n_estimators': [50, 100, 200, 400]})
```

In [33]:
```
y_pred2=model1.predict(X_test)
```

In [34]:
```
accuracy_score(y_test,y_pred2)
```

Out[34]:
```
1.0
```

In [35]:
```
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 [36]:
```
print(model1.best_estimator_)
```

```
GradientBoostingClassifier(n_estimators=50)
```

Step5. [Build AdaBoostClassifier]

In [37]:
```
base = DecisionTreeClassifier()
```

In [38]:
```
model2 = AdaBoostClassifier(base_estimator=base,random_state=0)
```

In [39]:
```
param_grid = {'n_estimators': [100, 150, 200], 'learning_rate': [0.01, 0.001]}
```

In [40]:
```
model3 = GridSearchCV(model2,param_grid,cv=5,n_jobs=-1)
```

In [42]:
```
model3.fit(X_train,y_train)
```

Out[42]:
```
GridSearchCV(cv=5,
             estimator=AdaBoostClassifier(base_estimator=DecisionTreeClassifi
er(),
                                          random_state=0),
             n_jobs=-1,
             param_grid={'learning_rate': [0.01, 0.001],
                         'n_estimators': [100, 150, 200]})
```

In [43]:
```
y_pred3=model3.predict(X_test)
```

In [44]:
```
accuracy_score(y_test,y_pred3)
```

Out[44]:
```
0.9977777777777778
```

In [45]:
```
print(classification_report(y_test,y_pred3))
```

```
              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
```

In [46]:
```
print(model3.best_estimator_)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), learning_rate=0.0
1,
                   n_estimators=100, random_state=0)
```

Step6. [Build LogisticRegressionCV classifier]

In [47]:
```
model4 = LogisticRegressionCV(cv=4,Cs=5,penalty='l2')
```

In [49]:
```
model4.fit(X_train,y_train)
```

Out[49]:
```
LogisticRegressionCV(Cs=5, cv=4)
```

In [50]:
```
y_pred4=model4.predict(X_test)
```

In [51]:
```
accuracy_score(y_test,y_pred4)
```

Out[51]:
```
0.9977777777777778
```

In [52]:
```
print(classification_report(y_test,y_pred4))
```

```
              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 [53]:
```
model5=VotingClassifier(estimators=[('lr',model4),('gbc',model1)], voting='har
```

In [54]:
```
model5.fit(X_train,y_train)
```

Out[54]:
```
VotingClassifier(estimators=[('lr', LogisticRegressionCV(Cs=5, cv=4)),
                             ('gbc',
                              GridSearchCV(cv=5,
                                           estimator=GradientBoostingClassifi
er(),
                                           n_jobs=-1,
                                           param_grid={'learning_rate': [0.1,
                                                                         0.0
1],
                                                       'n_estimators': [50, 1
00,
                                                                        200,
                                                                        40
0]}))])
```

In [55]:
```
y_pred5=model5.predict(X_test)
```

In [56]:
```
print(classification_report(y_test,y_pred5))
```

```
              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
```

Step8. [Interpret your results]

In [57]:
```
print(model1.best_estimator_)
```

```
GradientBoostingClassifier(n_estimators=50)
```

In [58]:
```
print(model3.best_estimator_)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), learning_rate=0.0
1,
                   n_estimators=100, random_state=0)
```

GradientBoostingClassifier

GradientBoostingClassifier(n_estimators=50)

In [59]:
```
classifierF = GradientBoostingClassifier(n_estimators=50)
all_scoresF = cross_val_score(estimator=classifier, X=X_train, y=y_train, cv=5
parameter = {'n_estimators': [50, 100, 200, 400], 'learning_rate': [0.1, 0.01]
```

In [60]:
```
modelGC = GridSearchCV(estimator=classifier, param_grid=parameter,cv=5, n_jobs
```

In [61]:
```
modelGC.fit(X_train,y_train)
```

Out[61]:
```
GridSearchCV(cv=5, estimator=GradientBoostingClassifier(), n_jobs=-1,
             param_grid={'learning_rate': [0.1, 0.01],
                         'n_estimators': [50, 100, 200, 400]})
```

In [62]:
```
y_predGC=model3.predict(X_test)
```

In [63]:
```
accuracy_score(y_test,y_predGC)
```

Out[63]:
```
0.9977777777777778
```

In [64]:
```
print(classification_report(y_test,y_predGC))
```

```
              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
```

AdaBoostClassifier

AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),learning_ random_state=0)

In [66]:
```
modelABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), learnin
```

In [67]:
```
param_grid = {'n_estimators': [100, 150, 200], 'learning_rate': [0.01, 0.001]}
```

In [68]:
```
modelGSCV = GridSearchCV(modelABC,param_grid,cv=5,n_jobs=-1)
modelGSCV.fit(X_train,y_train)
```

Out[68]:
```
GridSearchCV(cv=5,
             estimator=AdaBoostClassifier(base_estimator=DecisionTreeClassifi
er(),
                                          learning_rate=0.01, n_estimators=10
0,
                                          random_state=0),
             n_jobs=-1,
             param_grid={'learning_rate': [0.01, 0.001],
                         'n_estimators': [100, 150, 200]})
```

In [69]:
```
y_predGSCV=model3.predict(X_test)
```

In [70]:
```
accuracy_score(y_test,y_predGSCV)
```

Out[70]:
```
0.9977777777777778
```

In [71]:
```
print(classification_report(y_test,y_predGSCV))
```

```
              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
```

In [ ]: