Logistic Regression on Human activity recognition using Smartphones

Data collected form people carrying smartphones while performing activities with sensors. Activities are classified into walking , walking upstairs , walking downstairs, sitting , standing and laying .

Working on DataSet from Kaggle and Using Logistic Regression to predict type of activites.

Target : Activities to be classified into walking , walking upstairs , walking downstairs, sitting , standing and laying . Features : Sensors of smart phones .

In [76]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
```

In [77]:

```
data = pd.read_csv("data/Human_Activity_Recognition_Using_Smartphones_D
```

In [78]:

```
print(data.shape)
```

```
(10299, 562)
```

In [79]:

```
#Print no of integers, floats and strings
data.dtypes.value_counts()
```

Out[79]:

```
float64    561
object       1
dtype: int64
```

In [80]:

```
data.head()
```

Out[80]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBody ma
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.99
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.99
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.99
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.99
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.99

5 rows × 562 columns

In [81]:

```
#Checking value counts of each activites to check whether its balanced
data.Activity.value_counts()
```

Out[81]:

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

Preprocessing Steps

1. Select Features and convert target variable to int.
2. Split the data into train and test sets.

1. Select Features and Convert target variable to int.

In [82]:

```
feature_cols = data.columns[:-1]
#Encoding actitivity as an integer for scikit learn to process.
print(data['Activity'].dtypes)
```

```
object
```

In [83]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Activity'] = le.fit_transform(data.Activity)
```

2. Split Data to Train and Test sets

In [84]:

```
### StratifiedShuffleSplit is used to mainitan same ratio of predictor
from sklearn.model_selection import StratifiedShuffleSplit

# Get the split indexes
strat_shuf_split = StratifiedShuffleSplit(n_splits=1,
                                          test_size=0.3,
                                          random_state=42)

train_idx, test_idx = next(strat_shuf_split.split(data[feature_cols], d

# Create the dataframes
X_train = data.loc[train_idx, feature_cols]
y_train = data.loc[train_idx, 'Activity']

X_test  = data.loc[test_idx, feature_cols]
y_test  = data.loc[test_idx, 'Activity']
```

Modeling with Logisitc Regression

In [85]:

```
from sklearn.linear_model import LogisticRegression

# Standard logistic regression
lr = LogisticRegression(solver='liblinear').fit(X_train, y_train)
```

In [86]:

```
coeffs = lr.coef_
print(coeffs.T)
```

```
[[-0.09 -0.45  0.26 -0.17  0.76 -0.08]
 [ 0.01 -0.2   0.06 -0.02  0.11 -0.25]
 [ 0.03  0.06  0.25  0.02  0.06 -0.4 ]
 ...
 [ 1.6  -2.47 -0.85 -0.31 -0.71  0.42]
 [-0.37 -0.74  1.77 -0.24 -0.27  0.88]
 [-0.19 -0.3   0.42  0.07 -0.04  0.44]]
```

In [87]:

```
#making Predictions
y_hat = lr.predict(X_test)
```

In [88]:

```
# Use score method to get accuracy of model
score = lr.score(X_test, y_test)
print(score)
```

```
0.9841423948220065
```

Confusion Matrix

In [89]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_hat)
print(cm)
```

```
[[583   0   0   0   0   0]
 [  0 512  21   0   0   0]
 [  0  22 550   0   0   0]
 [  0   0   0 515   1   1]
 [  0   0   0   1 420   1]
 [  0   0   0   1   1 461]]
```

Confusion Matrix shows models ability to correctly predict or seperate classes .

Precison /Recall and F1- score

In [90]:

```
print (classification_report(y_test, y_hat))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       583
           1       0.96      0.96      0.96       533
           2       0.96      0.96      0.96       572
           3       1.00      1.00      1.00       517
           4       1.00      1.00      1.00       422
           5       1.00      1.00      1.00       463

    accuracy                           0.98      3090
   macro avg       0.98      0.98      0.98      3090
weighted avg       0.98      0.98      0.98      3090
```

Summary

Logistic Classifier could predict activities properly

Classifier is getting confused with sitting and standing