Human Activity Recognition using Smartphone Data with Machine Learning

Importing necessary libraries for the project

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib·inline
import warnings
warnings.filterwarnings("ignore")
```

Reading the data-set

```
train = pd.read_csv("train-1.csv")
test = pd.read_csv("test.csv")
```

```
train['Data'] = 'Train'
test['Data'] = 'Test'
both = pd.concat([train, test], axis=0).reset_index(drop=True)
both['subject'] = '#' + both['subject'].astype(str)
```

```
train.shape, test.shape
```

```
((7352, 564), (2947, 564))
```

```
both.head()
```

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBoc ma
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.9
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.9
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.9
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.9
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.9

5 rows × 564 columns

```
both.dtypes.value_counts()
```

```
float64    561
object       3
```

```
dtype: int64
```

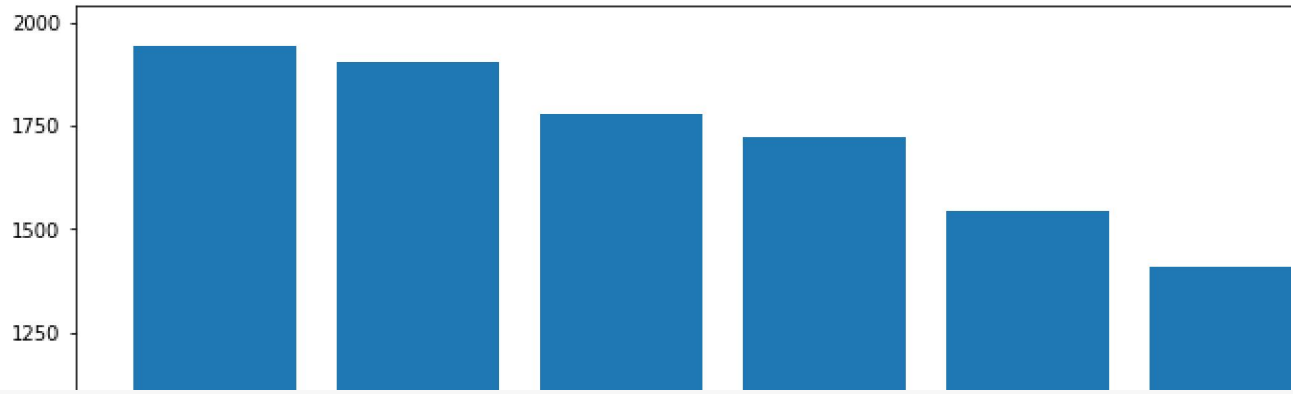
```
def basic_details(df):
    b = pd.DataFrame()
    b['Missing value'] = df.isnull().sum()
    b['N unique value'] = df.nunique()
    b['dtype'] = df.dtypes
    return b

basic_details(both)
```

	Missing value	N unique value	dtype
tBodyAcc-mean()-X	0	10292	float64
tBodyAcc-mean()-Y	0	10299	float64
tBodyAcc-mean()-Z	0	10293	float64
tBodyAcc-std()-X	0	10295	float64
tBodyAcc-std()-Y	0	10297	float64
...	...	...	...
angle(Y,gravityMean)	0	10299	float64
angle(Z,gravityMean)	0	10299	float64
subject	0	30	object
Activity	0	6	object
Data	0	2	object

564 rows × 3 columns

```
activity = both['Activity']
label_counts = activity.value_counts()

plt.figure(figsize= (12, 8))
plt.bar(label_counts.index, label_counts)
```

```
<BarContainer object of 6 artists>
```




```
Data = both['Data']
Subject = both['subject']
train = both.copy()
train = train.drop(['Data','subject','Activity'], axis =1)
```

Scaling the data

```
# Standard Scaler
from sklearn.preprocessing import StandardScaler
slc = StandardScaler()
train = slc.fit_transform(train)
```

```
# dimensionality reduction
from sklearn.decomposition import PCA
pca = PCA(n_components=0.9, random_state=0)
train = pca.fit_transform(train)
```

Splitting data for training and testing

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(train, activity, test_size = 0.2, rand
```

```
num_folds = 10
seed = 0
scoring = 'accuracy'
results = {}
accuracy = {}
```

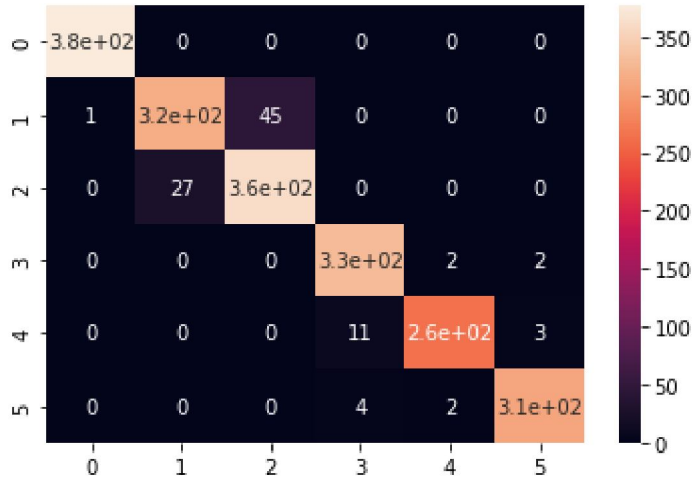
Algorithm to recognize human activity

```
# Finalizing the model and comparing the test, predict results
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.model_selection import KFold, cross_val_score
model = KNeighborsClassifier(algorithm= 'auto', n_neighbors= 8, p= 1, weights= 'distance')
```

```
_ = cross_val_score(model, X_train, y_train, cv=10, scoring=scoring)
results["GScv"] = (_.mean(), _.std())

model.fit(X_train, y_train)
y_predict = model.predict(X_test)

accuracy["GScv"] = accuracy_score(y_test, y_predict)

print(classification_report(y_test, y_predict))

cm= confusion_matrix(y_test, y_predict)
sns.heatmap(cm, annot=True)
```

```
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       377
           SITTING       0.92      0.87      0.90       364
          STANDING       0.89      0.93      0.91       390
           WALKING       0.96      0.99      0.97       335
WALKING_DOWNSTAIRS       0.99      0.95      0.97       278
  WALKING_UPSTAIRS       0.98      0.98      0.98       316

          accuracy                           0.95      2060
         macro avg       0.96      0.95      0.95      2060
      weighted avg       0.95      0.95      0.95      2060
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f22739fcfd0>
```



✓ 4s completed at 14:18