Human Activity Recognition from Smart Phone Data

Name - Amith Korada

Course - AI & ML
(Batch - 4)

Duration - 12 Months

Problem Statement - Perform activity recognition on the dataset

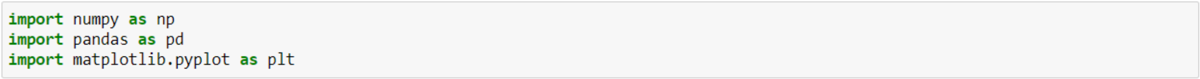
Prerequisites -

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has the latest version of python. The following URL https://www.python.org/downloads/ can be referred to as download python.

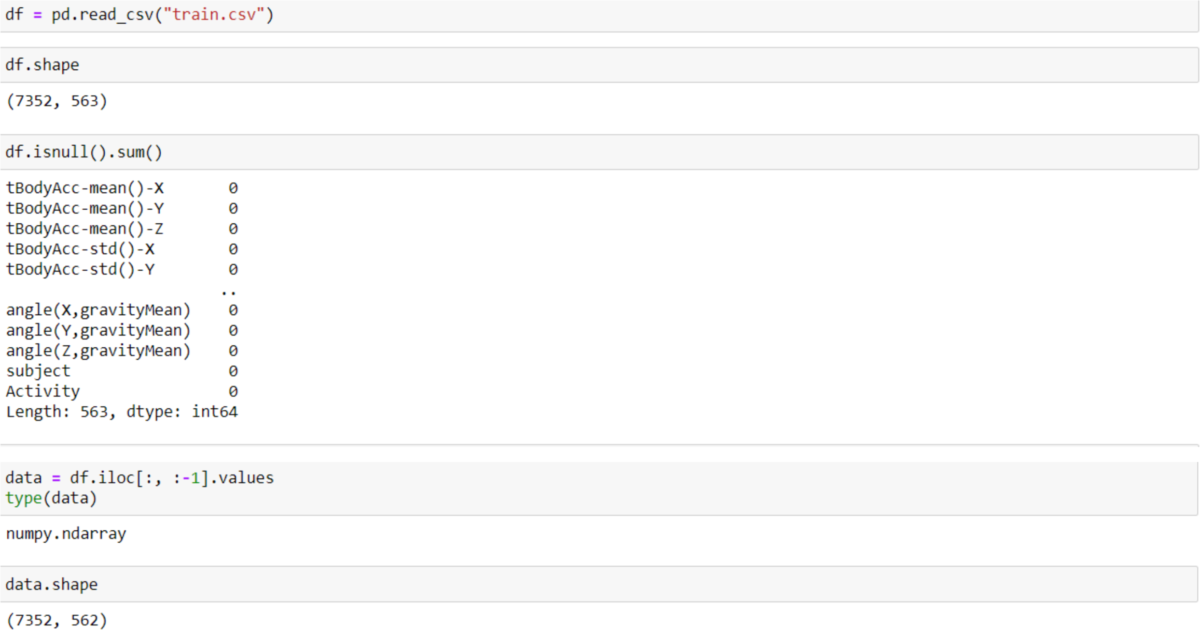
The second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this URL https://www.anaconda.com/download/ You will also need to download and install the below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run the below commands in command prompt/terminal to install these packages pip install -U sci-kit-learn pip install NumPy pip install scipy if you have chosen to install anaconda then run the below commands in anaconda prompt to install these packages conda install -c sci-kit-learn conda install -c anaconda numpy conda install -c anaconda scipy.

1. Importing necessary libraries-



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

2. Loading the dataset-



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

```
df.shape
```

```
(7352, 563)
```

```
df.isnull().sum()
```

```
tBodyAcc-mean()-X       0
tBodyAcc-mean()-Y       0
tBodyAcc-mean()-Z       0
tBodyAcc-std()-X        0
tBodyAcc-std()-Y        0
                       ..
angle(X,gravityMean)    0
angle(Y,gravityMean)    0
angle(Z,gravityMean)    0
subject                 0
Activity                0
Length: 563, dtype: int64
```

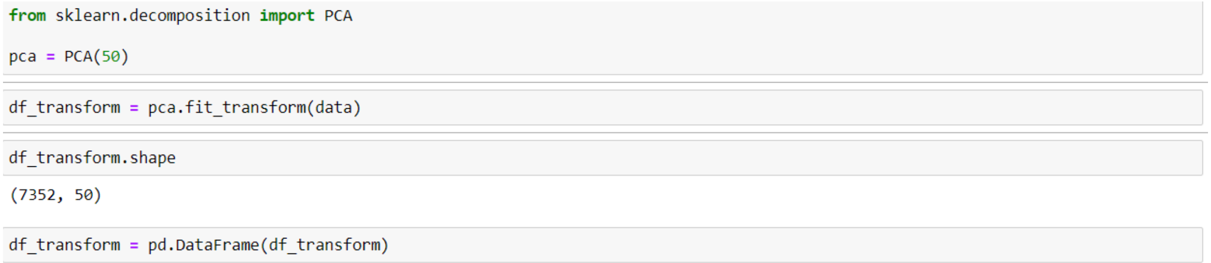
```
data = df.iloc[:, :-1].values
type(data)
```

```
numpy.ndarray
```

```
data.shape
```

```
(7352, 562)
```

3. Performing PCA for dimensionality reduction-



```
from sklearn.decomposition import PCA

pca = PCA(50)
```

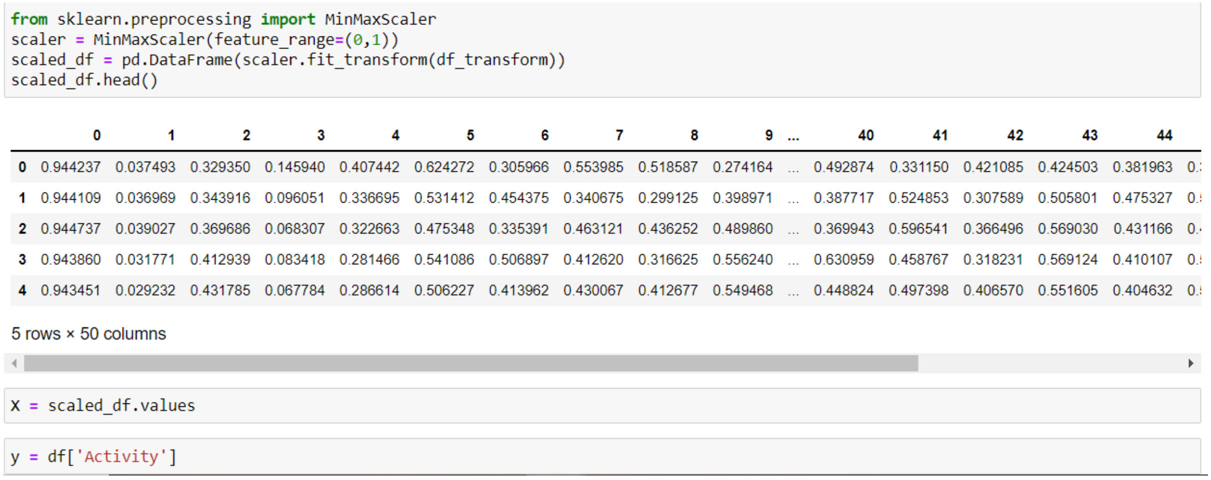
```
df_transform = pca.fit_transform(data)
```

```
df_transform.shape
```

```
(7352, 50)
```

```
df_transform = pd.DataFrame(df_transform)
```

4. Scaling the dataset-



```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
scaled_df = pd.DataFrame(scaler.fit_transform(df_transform))
scaled_df.head()
```

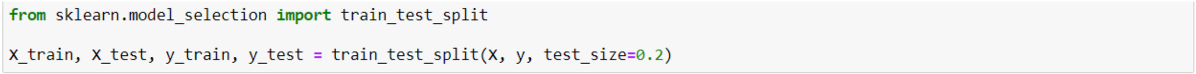
	0	1	2	3	4	5	6	7	8	9	...	40	41	42	43	44	
0	0.944237	0.037493	0.329350	0.145940	0.407442	0.624272	0.305966	0.553985	0.518587	0.274164	...	0.492874	0.331150	0.421085	0.424503	0.381963	0.
1	0.944109	0.036969	0.343916	0.096051	0.336695	0.531412	0.454375	0.340675	0.299125	0.398971	...	0.387717	0.524853	0.307589	0.505801	0.475327	0.
2	0.944737	0.039027	0.369686	0.068307	0.322663	0.475348	0.335391	0.463121	0.436252	0.489860	...	0.369943	0.596541	0.366496	0.569030	0.431166	0.
3	0.943860	0.031771	0.412939	0.083418	0.281466	0.541086	0.506897	0.412620	0.316625	0.556240	...	0.630959	0.458767	0.318231	0.569124	0.410107	0.
4	0.943451	0.029232	0.431785	0.067784	0.286614	0.506227	0.413962	0.430067	0.412677	0.549468	...	0.448824	0.497398	0.406570	0.551605	0.404632	0.

5 rows × 50 columns

```
X = scaled_df.values
```

```
y = df['Activity']
```

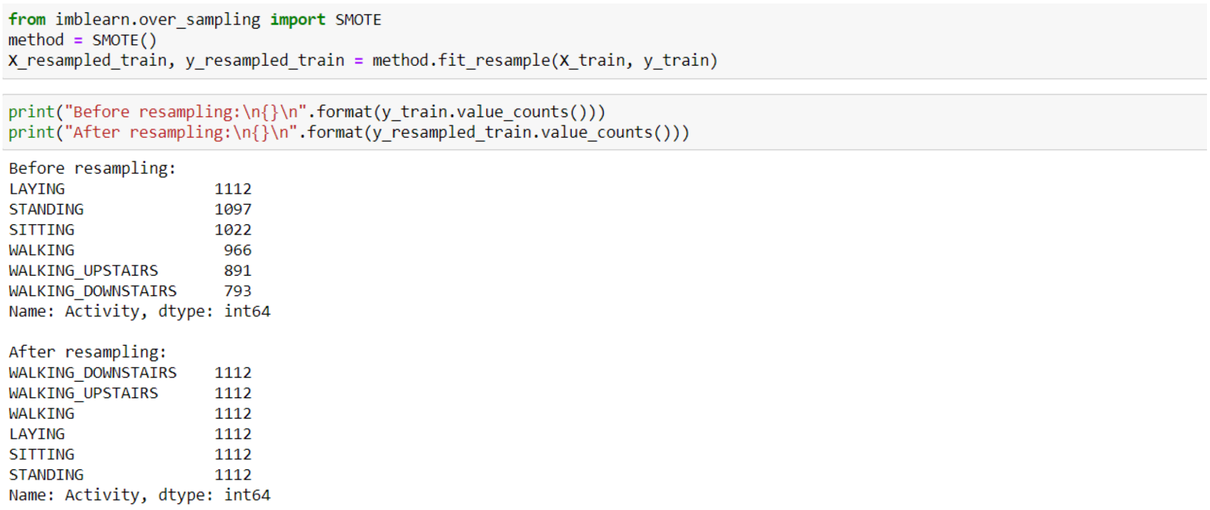
5. Splitting the dataset-



```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

6. Oversampling the data using SMOTE-



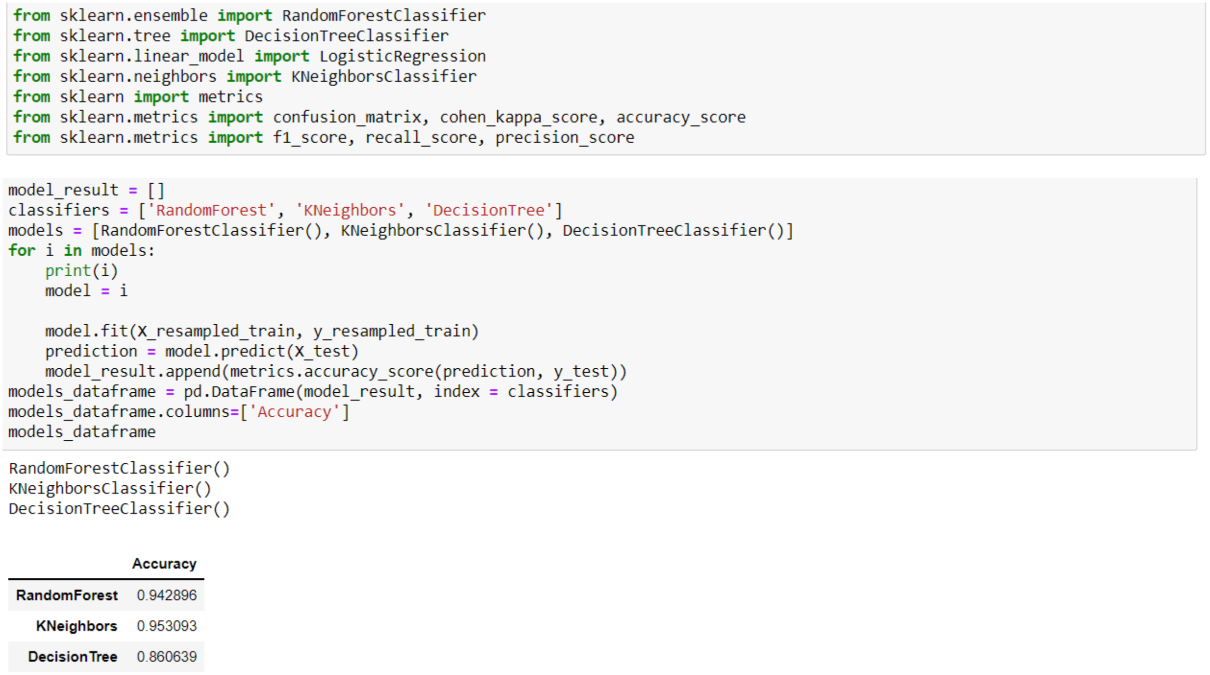
```
from imblearn.over_sampling import SMOTE
method = SMOTE()
X_resampled_train, y_resampled_train = method.fit_resample(X_train, y_train)
```

```
print("Before resampling:\n{}\n".format(y_train.value_counts()))
print("After resampling:\n{}\n".format(y_resampled_train.value_counts()))
```

```
Before resampling:
LAYING                1112
STANDING              1097
SITTING               1022
WALKING                966
WALKING_UPSTAIRS       891
WALKING_DOWNSTAIRS     793
Name: Activity, dtype: int64

After resampling:
WALKING_DOWNSTAIRS    1112
WALKING_UPSTAIRS      1112
WALKING               1112
LAYING                1112
SITTING               1112
STANDING              1112
Name: Activity, dtype: int64
```

7. Performing classification using different classifiers-



```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix, cohen_kappa_score, accuracy_score
from sklearn.metrics import f1_score, recall_score, precision_score
```

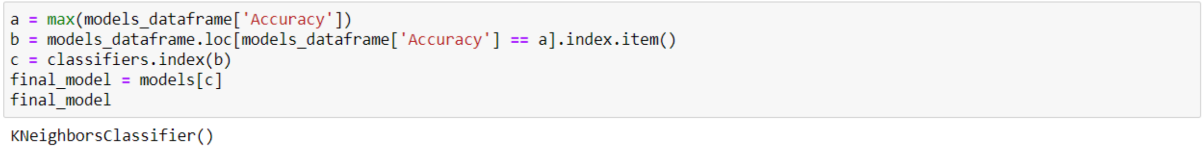
```
model_result = []
classifiers = ['RandomForest', 'KNeighbors', 'DecisionTree']
models = [RandomForestClassifier(), KNeighborsClassifier(), DecisionTreeClassifier()]
for i in models:
    print(i)
    model = i

    model.fit(X_resampled_train, y_resampled_train)
    prediction = model.predict(X_test)
    model_result.append(metrics.accuracy_score(prediction, y_test))
models_dataframe = pd.DataFrame(model_result, index = classifiers)
models_dataframe.columns=['Accuracy']
models_dataframe
```

```
RandomForestClassifier()
KNeighborsClassifier()
DecisionTreeClassifier()
```

	Accuracy
RandomForest	0.942896
KNeighbors	0.953093
DecisionTree	0.860639

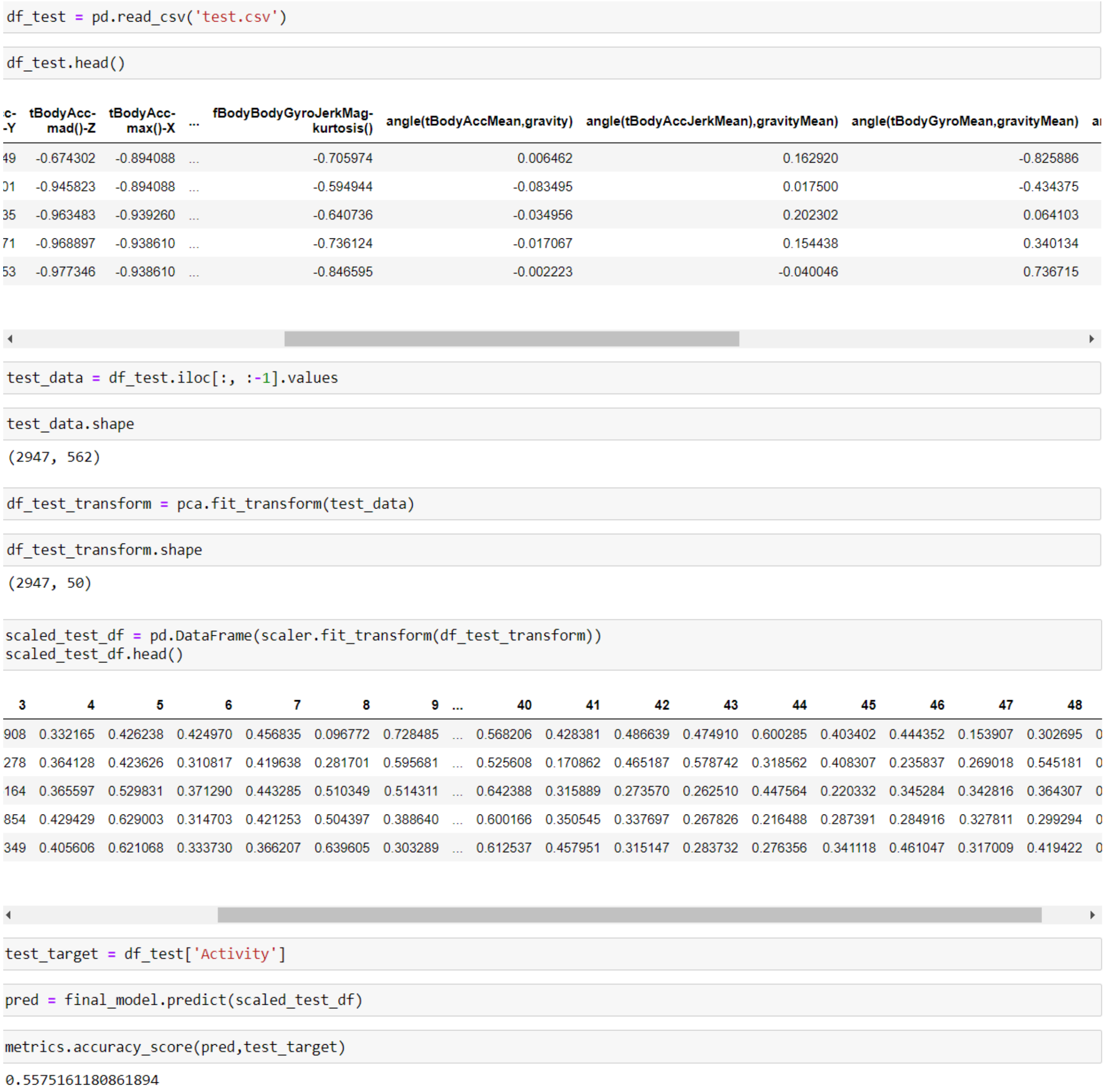
8. Selecting the model with the highest accuracy-



```
a = max(models_dataframe['Accuracy'])
b = models_dataframe.loc[models_dataframe['Accuracy'] == a].index.item()
c = classifiers.index(b)
final_model = models[c]
final_model
```

```
KNeighborsClassifier()
```

9. Loading the test data and predicting with the selected classifier-



```
df_test = pd.read_csv('test.csv')
```

```
df_test.head()
```

c- -Y	tBodyAcc- mad()-Z	tBodyAcc- max()-X	...	fBodyBodyGyroJerkMag- kurtosis()	angle(tBodyAccMean,gravity)	angle(tBodyAccJerkMean),gravityMean)	angle(tBodyGyroMean,gravityMean)	ai
49	-0.674302	-0.894088	...	-0.705974	0.006462	0.162920	-0.825886	
01	-0.945823	-0.894088	...	-0.594944	-0.083495	0.017500	-0.434375	
35	-0.963483	-0.939260	...	-0.640736	-0.034956	0.202302	0.064103	
71	-0.968897	-0.938610	...	-0.736124	-0.017067	0.154438	0.340134	
53	-0.977346	-0.938610	...	-0.846595	-0.002223	-0.040046	0.736715	

```
test_data = df_test.iloc[:, :-1].values
```

```
test_data.shape
```

```
(2947, 562)
```

```
df_test_transform = pca.fit_transform(test_data)
```

```
df_test_transform.shape
```

```
(2947, 50)
```

```
scaled_test_df = pd.DataFrame(scaler.fit_transform(df_test_transform))
scaled_test_df.head()
```

3	4	5	6	7	8	9	...	40	41	42	43	44	45	46	47	48	
908	0.332165	0.426238	0.424970	0.456835	0.096772	0.728485	...	0.568206	0.428381	0.486639	0.474910	0.600285	0.403402	0.444352	0.153907	0.302695	0
278	0.364128	0.423626	0.310817	0.419638	0.281701	0.595681	...	0.525608	0.170862	0.465187	0.578742	0.318562	0.408307	0.235837	0.269018	0.545181	0
164	0.365597	0.529831	0.371290	0.443285	0.510349	0.514311	...	0.642388	0.315889	0.273570	0.262510	0.447564	0.220332	0.345284	0.342816	0.364307	0
854	0.429429	0.629003	0.314703	0.421253	0.504397	0.388640	...	0.600166	0.350545	0.337697	0.267826	0.216488	0.287391	0.284916	0.327811	0.299294	0
349	0.405606	0.621068	0.333730	0.366207	0.639605	0.303289	...	0.612537	0.457951	0.315147	0.283732	0.276356	0.341118	0.461047	0.317009	0.419422	0

```
test_target = df_test['Activity']
```

```
pred = final_model.predict(scaled_test_df)
```

```
metrics.accuracy_score(pred,test_target)
```

```
0.5575161180861894
```