Report on Human Activity Recognition using Smartphones

P Ramakrishna Raju, M Nitin Sai, N Ravi Kiran, N V Siva Sai, N V Shanmukh

Amrita School of Engineering, Bengaluru
Foundations of Data Science

Abstract- The dataset which we have taken for this project, it is taken from a random 30 members. The six categories that smartphone is divided is walking, walking upstairs, walking downstairs, standing, sitting, laying. By using accelerometer, gforce and gyroscope we are detecting the movements of a human body.

Index Terms- activity, sensors, accelerometer, gyroscope, gforce, classifiers, kNN, graphs, relationship

I. Introduction

This article guides through various steps of data processing and analysis such as:

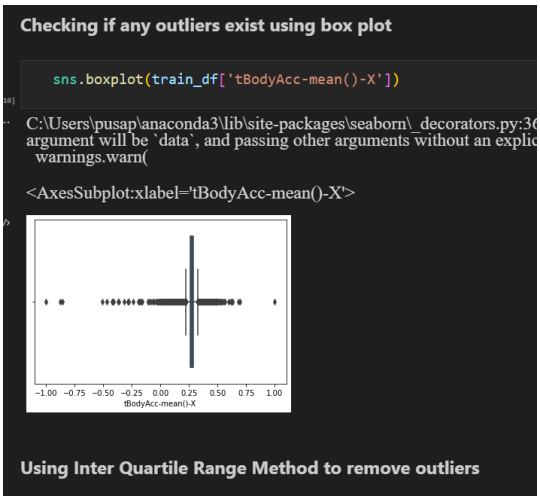
1) Data Cleaning
2) Data Visualization
3) Data Sampling
4) Feature extraction
5) Classification and accuracies
6) Conclusion

II. Data Cleaning

In data cleaning we have been able to remove noisy data, null values, outliers, duplicate values.

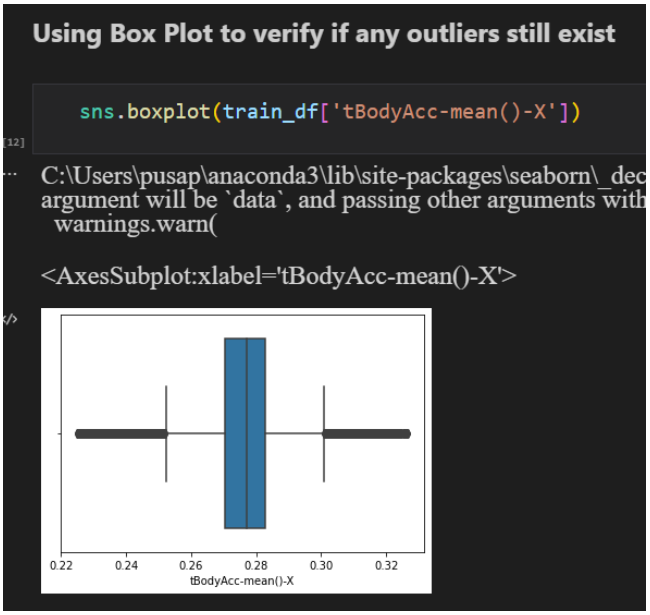
Checking for Null values:

Checking if any outliers exist using boxplot:



Removing outliers using inter-quartile range method:

Verifying if outliers still exist:



Checking for duplicate values :

```
duplicated = train_df[train_df.duplicated()]
```

```
duplicated
```

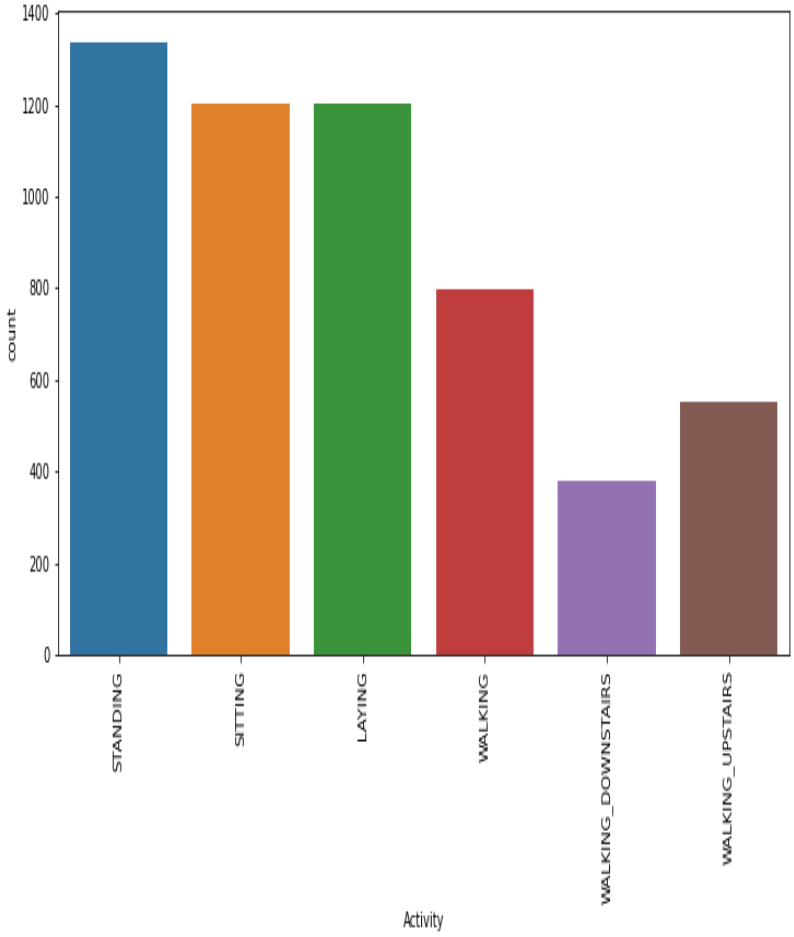
tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-kurtosis()

0 rows × 563 columns

III. DATA VISUALIZATION

Using count-plot we were able to see count of each activity :

We are able to identify the count of activities done by persons.

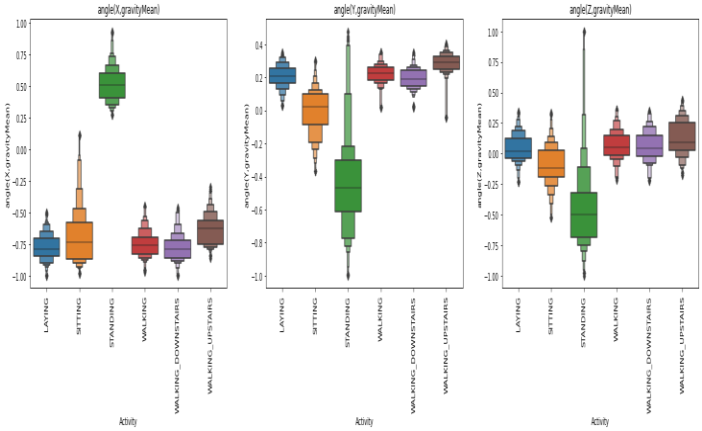


Using boxen-plot we were able to see:

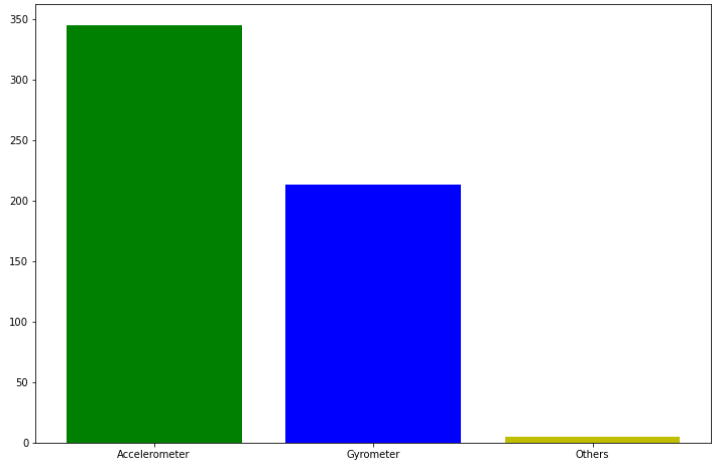
We can observe the skewness and kurtosis in each activity with features such as angles between sensors.

Mostly all activities follow normal distribution.

Comparatively STANDING activity is distributed widely from the below plot.



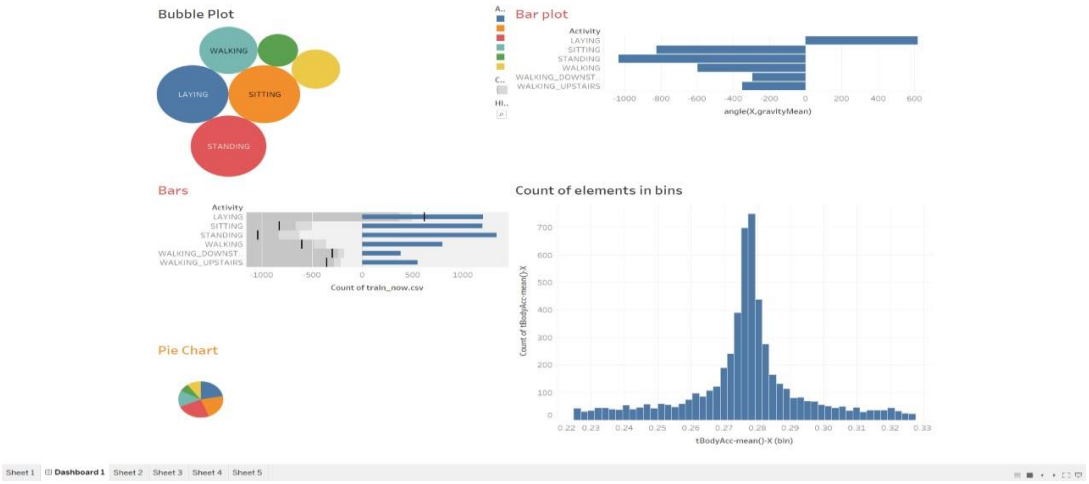
The following picture shows the number of features based on sensors:



Co-Variance matrix below shows most features are independent:

	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	...	tGravityAccMag-entropy()	tBody
tBodyAcc-mean()-X	1.000000	0.148061	-0.256952	0.000619	-0.021903	-0.044617	0.006290	-0.022754	-0.047558	0.044062	...	-0.031620	
tBodyAcc-mean()-Y	0.148061	1.000000	-0.078769	-0.045160	-0.044920	-0.049746	-0.044180	-0.045049	-0.050402	-0.038108	...	-0.026993	
tBodyAcc-mean()-Z	-0.256952	-0.078769	1.000000	-0.020217	-0.016641	-0.008410	-0.018747	-0.015203	-0.001988	-0.037197	...	-0.009533	
tBodyAcc-std()-X	0.000619	-0.045160	-0.020217	1.000000	0.927461	0.851668	0.998632	0.920888	0.846392	0.980844	...	0.877860	
tBodyAcc-std()-Y	-0.021903	-0.044920	-0.016641	0.927461	1.000000	0.895510	0.922803	0.997347	0.894509	0.917366	...	0.922191	
...	...	...	...	...	...	...	...	...	...	...	...	...	
tBodyGyroMag-min()	-0.053541	-0.021808	-0.043603	0.766412	0.790591	0.782720	0.760096	0.787366	0.781603	0.760145	...	0.777227	
tBodyGyroMag-sma()	-0.025803	-0.052827	-0.046097	0.916908	0.952195	0.935711	0.909555	0.947731	0.932718	0.916208	...	0.907855	
tBodyGyroMag-energy()	-0.019648	-0.053187	-0.061273	0.820725	0.853612	0.870072	0.813757	0.847208	0.861238	0.821490	...	0.754410	
tBodyGyroMag-iqr()	-0.012185	-0.050552	-0.041172	0.869272	0.914304	0.895621	0.861608	0.909592	0.893716	0.872753	...	0.868764	
tBodyGyroMag-entropy()	-0.017817	-0.001048	0.020986	0.500288	0.540858	0.485138	0.497411	0.543644	0.496098	0.495730	...	0.754158	

Tableau Visualization:



IV. Data Sampling

SAMPLING ALLOWS RESEARCHERS TO MAKE GENERALIZATIONS ABOUT A SPECIFIC POPULATION AND LEAVE OUT ANY BIAS

Simple Random Sampling:

Simple random sampling is a type of probability sampling in which the researcher randomly selects a subset of participants from a population.

```
train_df.sample(n=5, random_state=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	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-kurtosis()
3240	0.286415	0.000825	-0.106307	-0.990786	-0.967699	-0.964301	-0.992166	-0.970861	-0.961352	-0.930933	...	-0.923605
1199	0.275647	-0.013210	-0.109886	-0.984300	-0.932781	-0.945669	-0.985715	-0.937023	-0.946384	-0.929256	...	-0.652127
1678	0.278743	-0.016891	-0.106562	-0.993825	-0.990572	-0.993767	-0.995618	-0.989251	-0.992603	-0.922467	...	-0.935775
1166	0.279980	-0.009711	-0.130532	-0.994371	-0.956061	-0.973607	-0.995362	-0.953700	-0.972007	-0.935535	...	-0.882728
4401	0.285361	-0.000836	-0.098353	-0.994398	-0.939663	-0.973610	-0.995106	-0.942431	-0.975162	-0.935877	...	-0.190465

5 rows × 563 columns

Systematic Sampling:

Systematic sampling is a probability sampling method where researchers select members of the population at a regular interval.

Systematic Sampling

```
N = len(train_df)
n = 4
selected_index = np.arange(np.random.randint(0,n-1),len(train_df),N//n)
systematic_sampling = train_df.iloc[selected_index]
systematic_sampling
```

	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-kurtosis()
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104
1933	0.277937	-0.018022	-0.111468	-0.999004	-0.995342	-0.995509	-0.998879	-0.994713	-0.995258	-0.945219	...	-0.751307
3791	0.274067	-0.005486	-0.189612	0.003411	0.074264	-0.303971	-0.086013	0.095856	-0.358120	0.299966	...	-0.785095
5556	0.276528	-0.016754	-0.106263	-0.997451	-0.992599	-0.980884	-0.997546	-0.991357	-0.977349	-0.942241	...	-0.839537
7349	0.273387	-0.017011	-0.045022	-0.218218	-0.103822	0.274533	-0.304515	-0.098913	0.332584	0.043999	...	-0.304029

5 rows × 563 columns

Cluster Sampling:

Cluster sampling is a probability sampling method used by researchers in which you divide a population into clusters and then randomly select some of these clusters as your sample

Cluster Sampling

```
def get_clustered_Sample(df, n_per_cluster, num_select_clusters):
    N = len(df)
    K = int(N/n_per_cluster)
    data = None
    for k in range(K):
        sample_k = df.sample(n_per_cluster)
        sample_k["cluster"] = np.repeat(k,len(sample_k))
        df = df.drop(index = sample_k.index)
        data = pd.concat([data,sample_k],axis = 0)

    random_chosen_clusters = np.random.randint(0,K,size = num_select_clusters)
    samples = data[data.cluster.isin(random_chosen_clusters)]
    return(samples)

sample = get_clustered_Sample(df = train_df, n_per_cluster = 100, num_select_clusters = 2)
sample
```

	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	...	angle(tBodyAccMean,gravity)	angle(tBodyAccJerkMean),gravityMean)
5586	0.272631	-0.018432	-0.106812	-0.902521	-0.972792	-0.989427	-0.902604	-0.978147	-0.989797	-0.932896	...	0.062668	0.156403
2098	0.289066	-0.025510	-0.100030	-0.973195	-0.892363	-0.956805	-0.976737	-0.898093	-0.959126	-0.907610	...	-0.043591	0.174204
5895	0.276187	-0.011254	-0.106192	-0.997784	-0.991402	-0.978381	-0.998320	-0.992632	-0.975414	-0.943133	...	0.161558	0.074143
4736	0.277849	-0.018110	-0.111369	-0.998925	-0.993370	-0.990809	-0.999158	-0.992628	-0.991015	-0.944505	...	0.142809	0.073185
3057	0.277757	-0.020065	-0.094148	-0.997890	-0.984712	-0.990823	-0.998258	-0.985189	-0.990771	-0.942319	...	0.082901	-0.338448
...	...	...	...	...	...	...	...	...	...	...	...	...	...
3971	0.280789	-0.011038	-0.115636	-0.979039	-0.952266	-0.963711	-0.986032	-0.960204	-0.968329	-0.912973	...	-0.013597	-0.570739
4463	0.281496	-0.015050	-0.113915	-0.995964	-0.995171	-0.992375	-0.996648	-0.994879	-0.993697	-0.941203	...	0.109600	-0.201831
4924	0.278667	-0.016872	-0.109571	-0.996702	-0.991440	-0.992795	-0.997066	-0.992002	-0.995087	-0.939585	...	-0.179878	0.442362
4568	0.251195	-0.044567	-0.109595	-0.943126	-0.845201	-0.897039	-0.952878	-0.853975	-0.895071	-0.922185	...	0.401938	-0.003355
1857	0.259651	0.011960	-0.101016	-0.163937	0.428341	-0.043871	-0.174215	0.425737	-0.057738	-0.045514	...	0.269832	0.119188

200 rows × 564 columns

V. Feature Extraction

We have used an app called PhysiscsToolBoxSuite in an iPhone 11 to record the sensor values of accelerometer, gyroscope, gForce in all 3 axes with an interval of 0.011 seconds and then extracted them using properties like mean, median, min, max etc into 111 features by combining them into 2.56 second interval.

Input data from smartphone:

	time	gFx	gFy	gFz	ax	ay	az	wx	wy	wz
0	2022-12-29 8:20:44.8780	-0.126	-0.268	-0.985	0.15	-0.44	0.60	-0.25	-0.43	-0.07
1	2022-12-29 8:20:44.8870	-0.140	-0.256	-0.944	0.24	-0.49	0.13	-0.12	-0.39	-0.13
2	2022-12-29 8:20:44.8970	-0.154	-0.250	-0.921	0.31	-0.62	-0.18	0.04	-0.34	-0.19
3	2022-12-29 8:20:44.9070	-0.153	-0.256	-0.905	0.43	-0.57	-0.28	0.16	-0.28	-0.28
4	2022-12-29 8:20:44.9170	-0.152	-0.271	-0.895	0.29	-0.53	-0.48	0.22	-0.19	-0.34
...	...	...	...	...	...	...	...	...	...	...
1828	2022-12-29 8:21:03.0880	-0.047	-0.424	-0.970	0.52	-0.46	0.72	0.04	0.05	-0.04
1829	2022-12-29 8:21:03.0980	-0.013	-0.418	-0.966	0.30	-0.42	0.66	-0.05	-0.03	-0.05
1830	2022-12-29 8:21:03.1080	-0.009	-0.437	-1.024	0.13	-0.34	1.20	-0.17	-0.22	-0.05
1831	2022-12-29 8:21:03.1180	-0.016	-0.462	-0.996	0.14	-0.05	1.21	-0.24	-0.40	-0.02
1832	2022-12-29 8:21:03.1280	-0.032	-0.463	-0.939	0.26	0.10	0.74	-0.23	-0.46	0.01

Example of feature extraction using mean:

```
data_mean = new_data.groupby(pd.Grouper(key='time', freq='2.56S')).mean()
```

```
data_mean = data_mean.dropna()
```

```
data_mean
```

time	gFx	gFy	gFz	ax	ay	az	wx	wy	wz
2022-12-29 08:20:44.160	-0.047261	-0.308163	-0.934179	-0.082011	-0.083424	-0.038967	0.048967	-0.018641	-0.055761
2022-12-29 08:20:46.720	-0.064389	-0.398914	-0.908016	-0.003385	-0.070000	-0.041673	0.010156	0.000078	-0.007160
2022-12-29 08:20:49.280	-0.041872	-0.440844	-0.888747	-0.017393	-0.058833	-0.052724	0.019728	-0.005253	0.005486
2022-12-29 08:20:51.840	-0.035459	-0.430432	-0.894817	-0.010000	-0.058210	-0.046148	-0.003502	0.022412	-0.007704
2022-12-29 08:20:54.400	0.013514	-0.436533	-0.892366	-0.021868	-0.049027	-0.054047	0.008288	0.000156	-0.004825
2022-12-29 08:20:56.960	0.021949	-0.432097	-0.894681	-0.013541	-0.051868	-0.047704	0.003307	0.012374	0.003268
2022-12-29 08:20:59.520	0.053630	-0.448253	-0.884720	-0.007704	-0.060700	-0.051012	0.011556	-0.005603	-0.004786
2022-12-29 08:21:02.080	0.041849	-0.447670	-0.887226	-0.020566	-0.066226	-0.025283	-0.002170	-0.080849	-0.019340

Finally we were able to produce a final-set:

finalset

time	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	...	tGravityAccMag-entropy()	tBodyGyroMag-mean()	tBodyGyroMag-std()
2022-12-29 08:20:46.720	-0.003385	-0.070000	-0.041673	0.123898	0.106913	0.174499	0.084725	0.070584	0.119617	0.45	...	-0.931983	0.001305	0.136094
2022-12-29 08:20:49.280	-0.017393	-0.058833	-0.052724	0.105707	0.087085	0.146270	0.083656	0.069086	0.116296	0.38	...	-0.924035	0.014689	0.126799
2022-12-29 08:20:51.840	-0.010000	-0.058210	-0.046148	0.112767	0.080520	0.100103	0.083891	0.064498	0.077480	0.46	...	-0.926025	0.007015	0.097155
2022-12-29 08:20:54.400	-0.021868	-0.049027	-0.054047	0.131863	0.066092	0.105944	0.095288	0.049138	0.081211	0.43	...	-0.924704	0.001714	0.122242
2022-12-29 08:20:56.960	-0.013541	-0.051868	-0.047704	0.159947	0.123350	0.160136	0.119367	0.088937	0.122795	0.43	...	-0.926291	0.006301	0.175200
2022-12-29 08:20:59.520	-0.007704	-0.060700	-0.051012	0.187604	0.152695	0.151174	0.148081	0.121063	0.110355	0.58	...	-0.922422	0.000130	0.108974
2022-12-29 08:21:02.080	-0.020566	-0.066226	-0.025283	0.377334	0.341223	0.395513	0.261908	0.245116	0.266116	1.26	...	-0.923869	-0.064171	0.360034

VI. Classification and Accuracies

By training and testing using various models we finally have concluded with 3 models.
k-NN(k=10) classifier and Decision Tree Classifier since they gave us good accuracies 89% and 86% respectively.

SVM for bi-classification gave us 100% accuracy, where {LAYING, SITTING, STANDING} are Stationary activities and {WALKING, WALKING_DOWNSTAIRS, WALKING_UPSTAIRS} are moving activities.

res_time

	time	Activity
0	2022-12-29 08:20:46.720	LAYING
1	2022-12-29 08:20:49.280	LAYING
2	2022-12-29 08:20:51.840	LAYING
3	2022-12-29 08:20:54.400	LAYING
4	2022-12-29 08:20:56.960	LAYING
5	2022-12-29 08:20:59.520	LAYING
6	2022-12-29 08:21:02.080	LAYING

We were able to achieve good results from k-NN classifier.

VII. Conclusion

In conclusion we were able to understand the relationship between variables and sensors in a smartphone to understand how they work and precisely predict the activity done based on the sensor data using popular classifiers.

References

[1] Human Activity Recognition with Smartphones | Kaggle

[2] https://friends-04.web.app/