

Project Report
On
Human Activity Recognition
Submitted in partial fulfillment for the award
of **Post Graduate Diploma in Big Data**
Analytics
from **C-DAC (Noida)**

Guided by: Ms. Saruti Gupta

Presented by:

Mr. Anokha Vinay Tigga PRN: 230920525003

Mr. Mayank Agrawal PRN: 230920525009

**Centre of Development of Advanced Computing (C-DAC),
Noida**

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Mr. Anokha Vinay Tigga

Mr. Mayank Agrawal

have successfully completed their project on

HUMAN ACTIVITY RECOGNITION

under the guidance of Ms. Saruti Gupta

Program Head

Mr. Ravi Payal

Course Coordinator

Sidhidatri Nayak Mam

ACKNOWLEDGEMENT

This project “**Human Activity Recognition**” was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC NOIDA).

We all are very glad to mention the name of **Ms.Saruti** for her valuable guidance to work on this project. Her guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

We are highly grateful to Ms. Ravi Payal (Program Head (Noida training Centre), for her guidance and support whenever necessary while doing this course Diploma in **Big Data Analytics (DBDA)** through C-DAC NOIDA.

Our most heartfelt thanks goes to **Sidhidatri Nayak Mam** (Course Coordinator, **DBDA**) who gave all the required support and kind coordination to provide all the necessities where needed.

From:

Mr. Anokha Tigga

PRN: 230920525003

Mr. Mayank Agrawal

PRN: 230920525009

TABLE OF CONTENT

- 1. Abstract**
- 2. Introduction and Overview of Project**
- 3. Dataset Description**
- 4. Model and Big Pyspark Technologies Used**
- 5. Evaluation Metrics**
- 6. Procedure**
- 7. Models Used**
- 8. Prediction**
- 9. Conclusion**
- 10. Future Scope**
- 11. Bibliography**

1. ABSTRACT

The topic of human activity recognition is a prominent research area topic in the field of computer vision and image processing area. It has empowered state-of-art application in multiple sectors, surveillance, digital entertainment and medical healthcare.

It is interesting to observe and intriguing to predict such kind of movements. Several sensor-based approaches have also been introduced to study and predict human activities such accelerometer, gyroscope, etc., it has its own advantages and disadvantages.

In this project, an intelligent human activity recognition system is developed using Random Forest Classifier machine learning model.

2. INTRODUCTION AND OVERVIEW OF PROJECT

Human activity recognition using smartphone sensors like accelerometer is one of the hectic topics of research. In this project Random Forest Classifier machine learning model have been worked out to get the best final result.

In the same sequence, we use Random Forest Classifier model to recognize various activities of humans like 'biking', 'sitting', 'standing', 'walking', 'stair up' and 'stair down'.

Accelerometers detect magnitude and direction of the proper acceleration, as a vector quantity, and can be used to sense orientation (because direction of weight changes). GyroScope maintains orientation along a axis so that the orientation is unaffected by tilting or rotation of the mounting, according to the conservation of angular momentum.

Both the sensors generate data in 3D space over time. ('XYZ' represents 3-axial signals in X, Y, and Z directions.)

Objectives of the project:

Smart phones and other personal tracking devices used for fitness and health monitoring are cheap and ubiquitous. As such, sensor data from these devices is cheaper to collect, more common, and therefore is a more commonly studied version of the general activity recognition problem.

The problem is to predict the activity given a snapshot of sensor data, typically data from one or a small number of sensor types.

It is a challenging problem as there are no obvious or direct ways to relate the recorded sensor data to specific human activities and subject may perform an activity with significant variation, resulting in variations in the recorded sensor data.

The intent is to record sensor data and corresponding activities for specific subjects, fit a model from this data, and generalize the model to classify the activity of new unseen subjects from their sensor data.

3. DATASET DESCRIPTION

The heterogeneity dataset for human activity recognition from smartphone and smartwatches is a dataset devised to benchmark human activity recognition algorithms (classification, automatic data segmentation, sensor fusion, feature extraction, etc) containing sensor heterogeneities. The files in this archive contain all the samples from the activity recognition experiment. the dataset contains the readings of two motion sensors commonly found in smartphones' recorded while users executed activities scripted in no specific order carrying smartwatches and smartphones.

The data is split into 4 files in total divided by device (phone or watch) and sensor (gyroscope and accelerometer). The files for phones are: phones_accelerometer.csv, phones_gyroscope.csv for the accelerometer and gyroscope respectively, and for the watch_accelerometer.csv, watch_gyroscope.csv for the accelerometer and gyroscope as well. activities: 'biking', 'sitting', 'standing', 'walking', 'stair up' and 'stair down'. Sensors: two embedded sensors, i.e., accelerometer and gyroscope sampled at the highest frequency possible by the device Devices: 4 smartwatches (2 lg watches, 2 Samsung galaxy gears) 8 smartphones (2 Samsung galaxy s3 mini, 2 Samsung galaxy s3, 2 LG nexus 4, 2 Samsung galaxy s+) recordings: 9 users currently named: a, b, c, d, e, f, g, h, i consistently across all files.

The data set is structured in the following way:

----- Accelerometer samples -----

All the csv files have the same structure of following columns: 'index', 'arrival_time', 'creation_time', 'x', 'y', 'z', 'user', 'model', 'device', 'gt' And the columns have the following values:

Index: is the row number.

arrival_time: the time the measurement arrived to the sensing application

creation_time : the timestamp the os attaches to the sample

x,y,z : the values provided by the sensor for the three axes, x,y,z

User: the user this sample originates from, the users are named a to i.

Model: the phone/watch model this sample originates from

Device: the specific device this sample is from. They are prefixed with the model name and then the number, e.g., nexus4_1 or nexus4_2.

Gt: the activity the user was performing: bike sit, stand, walk, stairsup, stairsdown and null

Each accelerometer sample is represented as a single row in the file and with all columns having repeated values. Also due to issues with sampling some users have few collected samples for specific activities, e.g., user h and activity sit in the phones_accelerometer.csv.

4. MODEL AND PySpark TECHNOLOGIES USED

Random Forest Algorithm

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML.
- It is based on the concept of **ensemble learning**, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset
- Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

HIVE

- Hive is client software that convert Hive QL queries to MR. Hive QL is similar to SQL with many extended features. Hive manages structured data.
- Hive is data warehouse (OLAP) built for Hadoop. Apache Hive is a data warehouse implementation on top of Hadoop/HDFS.
- It provides SQL like interface to interact with the data stored in HDFS. It is not a complete database. It stores only metadata with it and the data is stored in HDFS only. So each query written by user is converted to MapReduce code which then interacts with HDFS
- Hive can be used as OLAP system. It is best suited for analytics, report generation and mining like operations in large data warehouse
- SQL query written in Hive is called as HiveQL or HQL

Py-Spark

- Apache Spark is written in Scala programming language. PySpark has been released in order to support the collaboration of Apache Spark and Python, it actually is a Python API for Spark.
- In addition, PySpark, helps you interface with Resilient Distributed Datasets (RDDs) in Apache Spark and Python programming language.
- PySpark features quite a few libraries for writing efficient programs. Furthermore, there are various external libraries that are also compatible, example – PySparkSQL, MLlib etc.
- PySparkSql - A PySpark library to apply SQL-like analysis on a huge amount of

structured or semi-structured data. We can also use SQL queries with PySparkSQL.

Streamlit

The trend of Data Science and Analytics is increasing day by day. From the data science pipeline, one of the most important steps is model deployment. We have a lot of options in python for deploying our model. Some popular frameworks are Flask and Django. But the issue with using these frameworks is that we should have some knowledge of HTML, CSS, and JavaScript. Keeping these prerequisites in mind, Adrien Treuille, Thiago Teixeira, and Amanda Kelly created “Streamlit”. Now using streamlit you can deploy any machine learning model and any python project with ease and without worrying about the frontend. Streamlit is very user-friendly

5.EVALUATION METRICS

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

1.PROCEDURE

To start the project open google colab install pySpark

1. Data loading and processing In pySpark

```
[3] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[4] path = '/content/drive/MyDrive/Watch_accelerometer.csv'

[5] df = spark.read.csv(path,header = True, inferSchema=True)

[6] df.show()
```

Index	Arrival_Time	Creation_Time	x	y	z	User	Model	Device	gt
0	1424696638740	27920678471000	-0.5650316	-9.572019	-0.61411273	a	gear	gear_1	stand
1	1424696638740	27920681910000	-0.83258367	-9.713276	-0.60693014	a	gear	gear_1	stand
2	1424696638740	27920692014000	-1.0181342	-9.935339	-0.54408234	a	gear	gear_1	stand
3	1424696638741	27920701983000	-1.2228385	-10.142437	-0.5662287	a	gear	gear_1	stand
4	1424696638741	27920711906000	-1.5771804	-10.480618	-0.40282443	a	gear	gear_1	stand
5	1424696638741	27920721675000	-2.1643584	-10.920552	-0.18375498	a	gear	gear_1	stand
6	1424696638741	27920731721000	-2.973	-11.063007	0.21188685	a	gear	gear_1	stand
7	1424696638741	27920743061000	-3.8881836	-11.08276	0.6847417	a	gear	gear_1	stand
8	1424696638742	27920751586000	-4.8919525	-10.890625	1.01574	a	gear	gear_1	stand
9	1424696638742	27920825873000	-12.600683	-7.674015	-1.1791444	a	gear	gear_1	stand
10	1424696638742	27920827227000	-9.214086	-4.5567646	0.2172738	a	gear	gear_1	stand
11	1424696638742	27920830887000	-9.214086	-4.5567646	0.2172738	a	gear	gear_1	stand
12	1424696638742	27920840675000	-9.240421	-4.104859	0.22325931	a	gear	gear_1	stand
13	1424696638743	27920853362000	-9.273342	-3.7295678	0.24061728	a	gear	gear_1	stand
14	1424696638897	27921589696000	-9.1668	-3.6703112	-0.729633	a	gear	gear_1	stand
15	1424696639000	27921599563000	-9.153033	-3.6056678	-0.7326257	a	gear	gear_1	stand
16	1424696639001	27921609644000	-9.1512375	-3.6122518	-0.74519527	a	gear	gear_1	stand
17	1424696639001	27921619491000	-9.202713	-3.6463692	-0.729633	a	gear	gear_1	stand
18	1424696639001	27921629562000	-9.288904	-3.7361517	-0.7266402	a	gear	gear_1	stand
19	1424696639001	27921639447000	-9.2907	-3.7720647	-0.7302315	a	gear	gear_1	stand

only showing top 20 rows

This snap shows loading of data in google colab Pyspark

2.Data Cleaning by deleting Null values

```
print('No. of rows are : ',df.count())
print('No of columns are : ',len(df.columns))
```

```
No. of rows are : 3540962
No of columns are : 10
```

```
df.na.drop().show()
```

Index	Arrival_Time	Creation_Time	x	y	z	User	Model	Device	gt
0	1424696638740	27920678471000	-0.5650316	-9.572019	-0.61411273	a	gear	gear_1	stand
1	1424696638740	27920681910000	-0.83258367	-9.713276	-0.60693014	a	gear	gear_1	stand
2	1424696638740	27920692014000	-1.0181342	-9.935339	-0.54408234	a	gear	gear_1	stand
3	1424696638741	27920701983000	-1.2228385	-10.142437	-0.5662287	a	gear	gear_1	stand
4	1424696638741	27920711906000	-1.5771804	-10.480618	-0.40282443	a	gear	gear_1	stand
5	1424696638741	27920721675000	-2.1643584	-10.920552	-0.18375498	a	gear	gear_1	stand
6	1424696638741	27920731721000	-2.973	-11.063007	0.21188685	a	gear	gear_1	stand
7	1424696638741	27920743061000	-3.8881836	-11.08276	0.6847417	a	gear	gear_1	stand
8	1424696638742	27920751586000	-4.8919525	-10.890625	1.01574	a	gear	gear_1	stand
9	1424696638742	27920825873000	-12.600683	-7.674015	-1.1791444	a	gear	gear_1	stand
10	1424696638742	27920827227000	-9.214086	-4.5567646	0.2172738	a	gear	gear_1	stand
11	1424696638742	27920830887000	-9.214086	-4.5567646	0.2172738	a	gear	gear_1	stand
12	1424696638742	27920840675000	-9.240421	-4.104859	0.22325931	a	gear	gear_1	stand
13	1424696638743	27920853362000	-9.273342	-3.7295678	0.24061728	a	gear	gear_1	stand
14	1424696638897	27921589696000	-9.1668	-3.6703112	-0.729633	a	gear	gear_1	stand
15	1424696639000	27921599563000	-9.153033	-3.6056678	-0.7326257	a	gear	gear_1	stand
16	1424696639001	27921609644000	-9.1512375	-3.6122518	-0.74519527	a	gear	gear_1	stand
17	1424696639001	27921619491000	-9.202713	-3.6463692	-0.729633	a	gear	gear_1	stand
18	1424696639001	27921629562000	-9.288904	-3.7361517	-0.7266402	a	gear	gear_1	stand
19	1424696639001	27921639447000	-9.2907	-3.7720647	-0.7302315	a	gear	gear_1	stand

only showing top 20 rows

3.Snap shows numbers of row and column

```
7] df.write.option("header","true").partitionBy('User').mode("overwrite").csv('/content/drive/MyDrive/data_partition_output')
```

```
8] df3 = spark.read.option('header','true').csv('/content/drive/MyDrive/data_partition_output/User=b')
```

```
1] df3.show()
```

Index	Arrival_Time	Creation_Time	x	y	z	Model	Device	gt
0	1424784652666	16203264658000	0.24181437	5.681441	7.5848308	gear	gear_1	stand
1	1424784652666	16203274131000	0.31423897	5.713164	7.4854717	gear	gear_1	stand
2	1424784652666	16203284818000	0.22804771	5.6407394	7.5770497	gear	gear_1	stand
3	1424784652666	16203293964000	0.22685061	5.6084175	7.6428905	gear	gear_1	stand
9	1424784652667	16203402696000	0.85413146	-8.988432	3.470994	gear	gear_1	stand
7	1424784652667	16203382788000	0.8170213	-8.917803	3.4554317	gear	gear_1	stand
8	1424784652667	16203392278000	0.82719666	-8.953117	3.4889505	gear	gear_1	stand
5	1424784652667	16203375476000	0.4938041	-8.765172	3.7110126	gear	gear_1	stand
4	1424784652667	16203304324000	0.2693477	5.651513	7.643489	gear	gear_1	stand
6	1424784652667	16203378518000	0.8170213	-8.917803	3.4554317	gear	gear_1	stand
10	1424784652668	16203412175000	0.87567925	-9.038112	3.5655649	gear	gear_1	stand
11	1424784652668	16203422589000	0.8798691	-9.094974	3.582923	gear	gear_1	stand
12	1424784652668	16203435934000	0.8714894	-9.122507	3.5727475	gear	gear_1	stand
13	1424784652668	16203451551000	0.8595184	-9.165004	3.6218286	gear	gear_1	stand
14	1424784652668	16203791204000	9.403227	-1.8489223	1.4173675	gear	gear_1	stand
18	1424784652669	16203831109000	9.41879	-1.8309658	1.4191631	gear	gear_1	stand
17	1424784652669	16203820719000	9.406818	-1.822586	1.4443022	gear	gear_1	stand
19	1424784652669	16203840475000	9.424775	-1.8602947	1.3988123	gear	gear_1	stand
15	1424784652669	16203800779000	9.387665	-1.834557	1.4335283	gear	gear_1	stand
16	1424784652669	16203811202000	9.387665	-1.8297687	1.4490906	gear	gear_1	stand

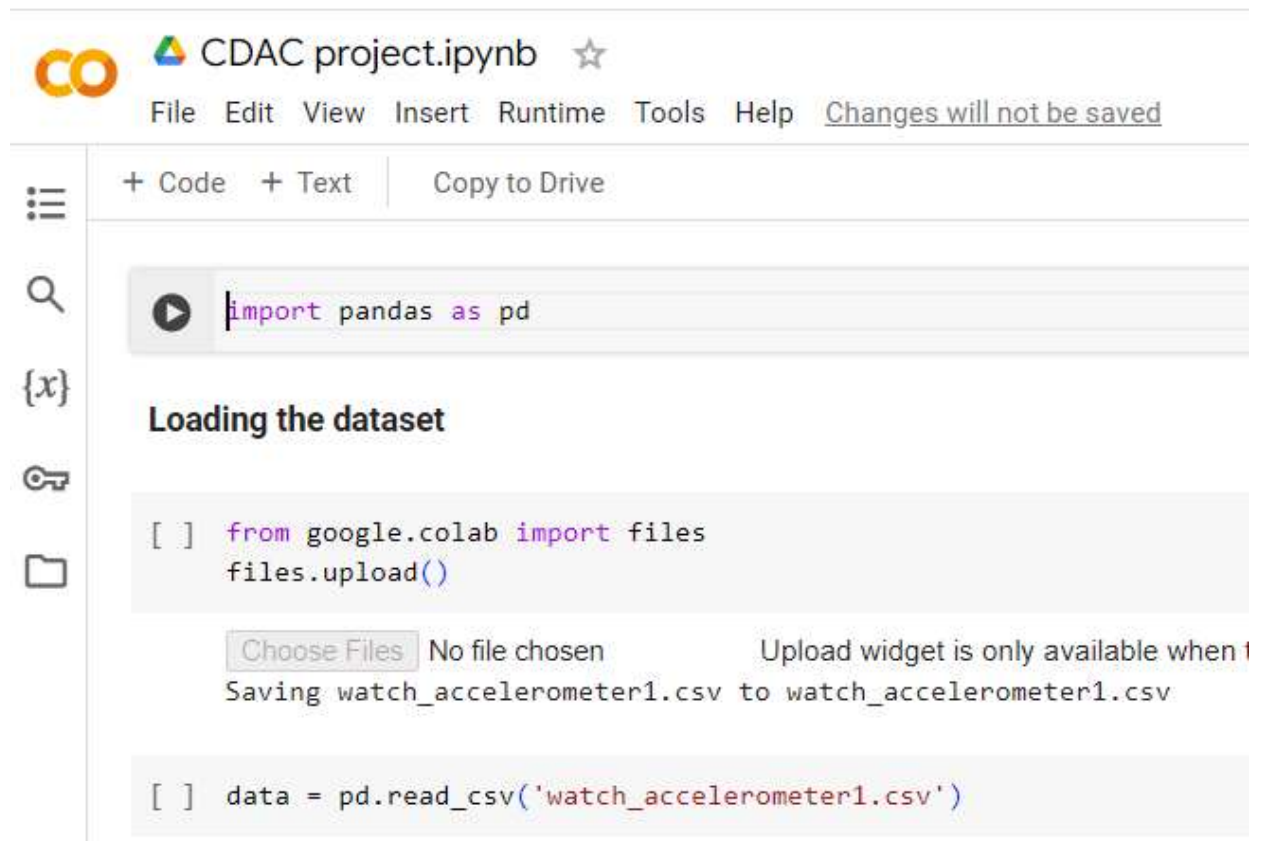
only showing top 20 rows

3.Snap shows files created by the pySpark Partitioning



Other Machine Learning Technology Used for Accuracy Check

1.Snap shows files upload file using Pandas Python library



The screenshot displays a Google Colab notebook interface. At the top, the title bar shows the Colab logo, the notebook name "CDAC project.ipynb", and a star icon. Below the title bar is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a status message "Changes will not be saved". On the left side, there is a sidebar with icons for a menu, search, variables, keys, and files. The main area of the notebook contains three code cells. The first cell is a code input cell with the text `import pandas as pd`. The second cell is a text cell titled "Loading the dataset" containing the code `from google.colab import files` and `files.upload()`. Below this code, there is a file upload widget with a "Choose Files" button, the text "No file chosen", and a message "Upload widget is only available when t". Below the widget, it says "Saving watch_accelerometer1.csv to watch_accelerometer1.csv". The third cell is a code input cell with the text `data = pd.read_csv('watch_accelerometer1.csv')`.

```
import pandas as pd
```

Loading the dataset

```
[ ] from google.colab import files
    files.upload()
```

Choose Files No file chosen Upload widget is only available when t
Saving watch_accelerometer1.csv to watch_accelerometer1.csv

```
[ ] data = pd.read_csv('watch_accelerometer1.csv')
```

2.Snap shows dataframe of dataset

```
[ ] data
```

	index	Arrival_Time	x	y	z	model	device	activity
0	0	2015-02-23 18:33:53	-9.160782	-3.759674	1.396469	lgwatch	lgwatch_1	stand
1	1	2015-02-23 18:33:53	-9.198868	-3.788238	1.420273	lgwatch	lgwatch_1	stand
2	2	2015-02-23 18:33:53	-9.208389	-3.804901	1.439316	lgwatch	lgwatch_1	stand
3	3	2015-02-23 18:33:53	-9.210770	-3.759674	1.446457	lgwatch	lgwatch_1	stand
4	4	2015-02-23 18:33:53	-9.222672	-3.735870	1.377426	lgwatch	lgwatch_1	stand
...
446209	381238	2015-02-23 19:05:17	-7.139847	-3.269318	4.702805	lgwatch	lgwatch_1	bike
446210	381239	2015-02-23 19:05:17	-6.485245	-3.545441	4.579025	lgwatch	lgwatch_1	bike
446211	381240	2015-02-23 19:05:17	-6.087723	-3.847748	4.886093	lgwatch	lgwatch_1	bike
446212	381241	2015-02-23 19:05:17	-6.049637	-3.885834	5.119370	lgwatch	lgwatch_1	bike
446213	381242	2015-02-23 19:05:17	-6.216263	-3.750153	5.124130	lgwatch	lgwatch_1	bike

3.Snap shows Information of dataset

```
▶ data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 446214 entries, 0 to 446213
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   index           446214 non-null  int64
1   Arrival_Time    446214 non-null  object
2   x               446214 non-null  float64
3   y               446214 non-null  float64
4   z               446214 non-null  float64
5   model           446214 non-null  object
6   device          446214 non-null  object
7   activity        446214 non-null  object
dtypes: float64(3), int64(1), object(4)
memory usage: 27.2+ MB
```

4.Snap shows Date and to ISD time dataset

```
data['Arrival_Time'] = data['Arrival_Time'].astype('datetime64[ns]')
data['year'] = data['Arrival_Time'].dt.year
data['month'] = data['Arrival_Time'].dt.month
data['day'] = data['Arrival_Time'].dt.day
data['hour'] = data['Arrival_Time'].dt.hour
data['minute'] = data['Arrival_Time'].dt.minute
data['second'] = data['Arrival_Time'].dt.second
```

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 446214 entries, 0 to 446213
Data columns (total 14 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   index           446214 non-null  int64
1   Arrival_Time    446214 non-null  datetime64[ns]
2   x               446214 non-null  float64
```

5.Snap shows description of dataset

+ Code + Text Copy to Drive

```
[ ] data.describe()
```

	index	x	y	z
count	446214.000000	446214.000000	446214.000000	446214.000000
mean	150657.279151	-8.490301	-2.579446	1.699798
std	120450.103283	2.941576	3.563373	2.881399
min	0.000000	-19.669525	-19.703400	-19.613300
25%	37870.250000	-9.434525	-4.959381	-0.129349
50%	117004.500000	-8.118179	-3.212189	1.386948
75%	255747.750000	-7.137466	-1.386841	2.656816
max	381242.000000	13.869308	19.612701	19.927063

```
[ ] data.shape
```

```
(446214, 8)
```


6.Snap shows number of rows and column

And checking of null values

```
▶ print('no. of rows :', data.shape[0])  
   print('no of columns : ', data.shape[1])
```

```
👤 no. of rows : 446214  
   no of columns : 8
```

Checking Null Values

```
[ ] data.isnull().sum()
```

```
index          0  
Arrival_Time   0  
x              0  
y              0  
z              0  
model          0  
device         0  
activity       0  
dtype: int64
```

7.Snap shows checking rows and column

Checking for duplicates value

```
[ ] data.duplicated().any().sum()
```

0

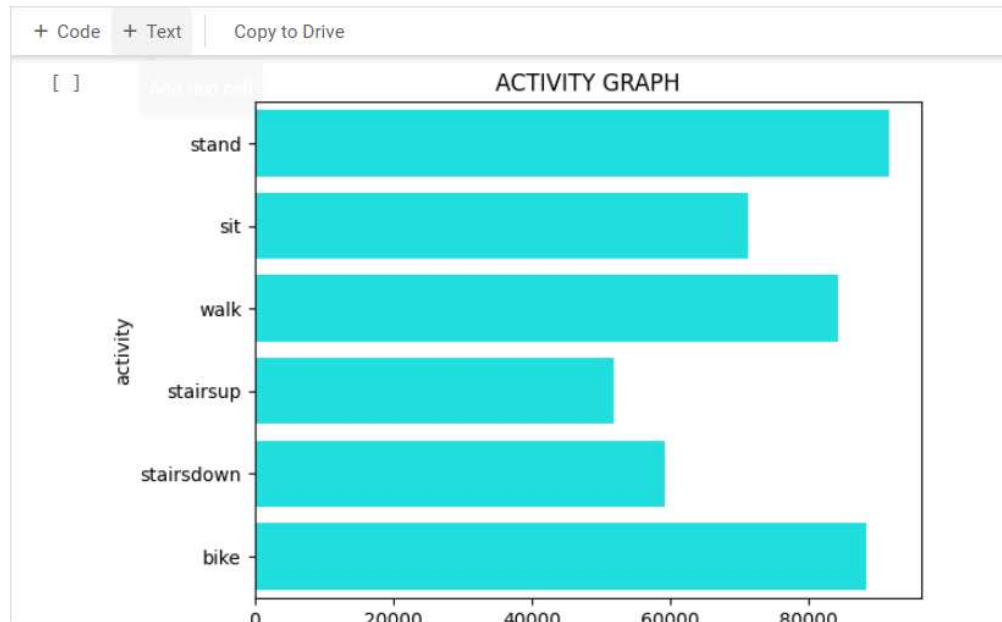
```
▶ import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt
```

```
[ ] data['activity'].value_counts()
```

```
stand      91588  
bike       88202  
walk       84260  
sit         71109  
stairsdown 59226  
stairsup   51829  
Name: activity, dtype: int64
```

Snap shows distinct activity count

8.Snap shows graph of activity



9.Snap shows LogisticRegression accuracy

```
[ ] log = LogisticRegression()  
log.fit(X_train,Y_train)
```

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(  
    LogisticRegression()  
    LogisticRegression()
```

```
[ ] y_pred1 = log.predict(X_test)  
LR_accuracy = accuracy_score(Y_test,y_pred1)  
LR_accuracy
```

0.8008807413466602

10.Snap shows Random Forest classifier accuracy

Random Forest

```
from sklearn.ensemble import RandomForestClassifier  
rf = RandomForestClassifier()  
rf.fit(X_train, Y_train)
```

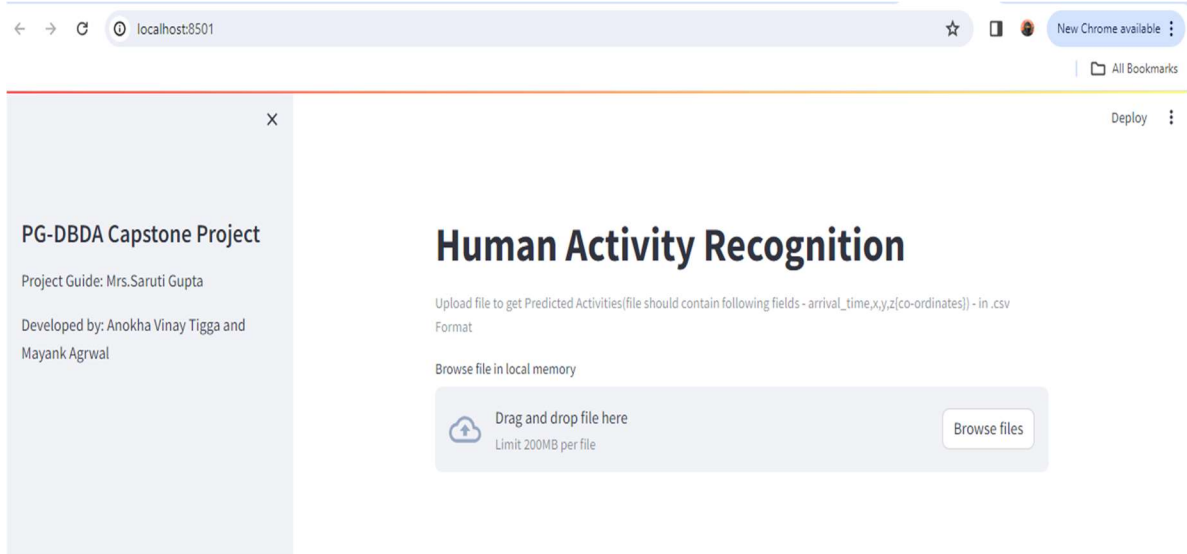


```
▼ RandomForestClassifier  
RandomForestClassifier()
```

```
[ ] y_pred2 = rf.predict(X_test)  
    RF_accuracy = accuracy_score(Y_test,y_pred2)  
    RF_accuracy
```

```
0.9994957587709961
```

- i. Processing dataset file by converting arrival_time column in yy/mm/dd hour/minutes/seconds format from unix epoch time then selecting features and dividing the dataset into train and test file before passing the data for training.



- ii. Selecting the Watch_acc3_test file for making predictions from localhost and processing it for before making predictions by changing arrival_time column format and selecting features

Human Activity Recognition

Upload file to get Predicted Activities(file should contain following fields - arrival_time,x,y,z(co-ordinates)) - in .csv Format

Browse file in local memory



Drag and drop file here
Limit 200MB per file

Browse files

 Watch_accelerometer.csv 44.0KB ×

	Arrival_Time	x	y	z
8	23-02-2015 18:33	-4.892	-10.8906	1.0157
9	23-02-2015 18:33	-12.6007	-7.674	-1.1791
10	23-02-2015 18:33	-9.2141	-4.5568	0.2173
11	23-02-2015 18:33	-9.2141	-4.5568	0.2173

- iii. GoogleAuthentication for Api services and accessing ganted.

← → ↻ ⓘ localhost:8080/?code=4%2F0AeaYSHCQqSVnAbqud2Hoa9GiqdyeBPGYlh5dWLCUX0sXbi

The authentication flow has completed.

- iv. Predictions Using Random Forest classifier .

The predictions are based on Random Forest Classifier ML model .

Accuracy Score:

0.9993949127852687

ROC_AUC Score:

0.9999973475397598

Model Training Complete

Activities predicted successfully

- v. final predictions for user.

Predictions

	x	y	z	Date	Time	Predicted_Activity
0	-0.565	-9.572	-0.6141	2015-2-23	18-33-0	stand
1	-0.8326	-9.7133	-0.6069	2015-2-23	18-33-0	stand
2	-1.0181	-9.9353	-0.5441	2015-2-23	18-33-0	stand
3	-1.2228	-10.1424	-0.5662	2015-2-23	18-33-0	stand
4	-1.5772	-10.4806	-0.4028	2015-2-23	18-33-0	stand
5	-2.1644	-10.9206	-0.1838	2015-2-23	18-33-0	stand
6	-2.973	-11.063	0.2119	2015-2-23	18-33-0	stand
7	-3.8882	-11.0828	0.6847	2015-2-23	18-33-0	stand
8	-4.892	-10.8906	1.0157	2015-2-23	18-33-0	stand
9	-12.6007	-7.674	-1.1791	2015-2-23	18-33-0	stand

- i. Displaying predicted score by Random Forest Classifier model.

Select Data Analysis Option from Sidebar

Execute ML Operation

[Click for Predictions](#)

Training dataset retrieved

The predictions are based on Random Forest Classifier ML model .

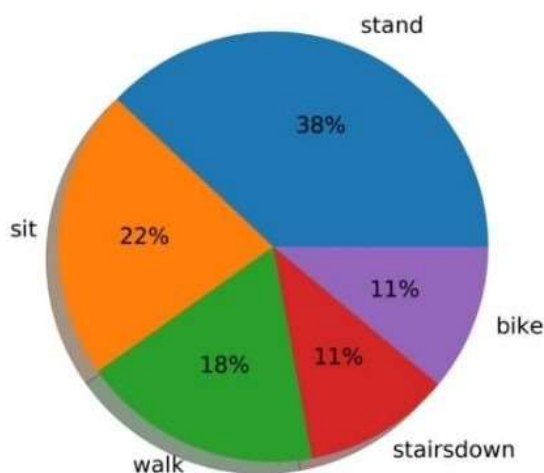
Accuracy Score:

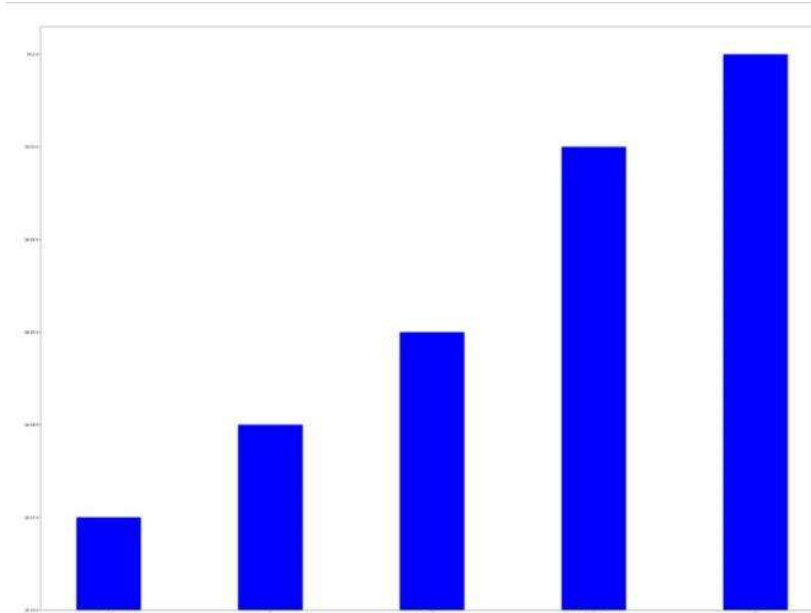
0.9993949127852687

ROC_AUC Score:

0.9999973475397598

- ii. Displaying Pie chart for predicted activities of the user and also plotting Bar chart with time on Y axis and activities on X axis for viewer to compare which activity is performed most and least.





iii. Displaying the predictions table.

PG-DBDA Capstone Project

Project Guide: Ms. Manasi Yeole

Developed by: Avinash Prakhar Rahul Shivam Mayuresh

Data Analysis

☐ none

☒ Charts

Share ☆ ☰

Predictions in Tabular Form

	x	y	z	Date	Time	Predicted_Activity
21	-9.1301	-3.1391	-0.1133	2015-2-23	18-33-0	stand
22	-9.1279	-3.6703	-0.8278	2015-2-23	18-33-0	stand
23	-9.1961	-3.6266	-0.8523	2015-2-23	18-33-0	stand
24	-9.2362	-3.7128	-0.8039	2015-2-23	18-33-0	stand
25	-9.2009	-3.6847	-0.7153	2015-2-23	18-33-0	stand
26	-9.1488	-3.6506	-0.6919	2015-2-23	18-33-0	stand
27	-9.1524	-3.6170	-0.6829	2015-2-23	18-33-0	stand
28	-9.2219	-3.7014	-0.7781	2015-2-23	18-33-0	stand
29	-9.3146	-3.6117	-0.7338	2015-2-23	18-33-0	stand
30	-9.3500	-3.5572	-0.7548	2015-2-23	18-33-0	stand

7. MODELS USED

Over the course of building the best model we tried some of the following machine learning algorithms which have been displayed below along with their accuracy score.

Logistic Regression

```
[ ] log = LogisticRegression()
    log.fit(X_train,Y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Con
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
    n_iter_i = _check_optimize_result(
    ▾ LogisticRegression
    LogisticRegression()

[ ] y_pred1 = log.predict(X_test)
    LR_accuracy = accuracy_score(Y_test,y_pred1)
    LR_accuracy

0.8008807413466602
```

By using this model we get the accuracy score as: 0.800

Decision Tree Classifier

DECISION TREE

```
▶ from sklearn.tree import DecisionTreeClassifier
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Initialize the decision tree classifier
clf = DecisionTreeClassifier()

# Train the classifier on the training data
clf.fit(X_train, Y_train)

# Make predictions on the testing data
y_pred3 = clf.predict(X_test)

# Calculate accuracy
DT_accuracy = accuracy_score(y_test, y_pred3)
DT_accuracy
```

0.8008807413466602

By using this model we get the accuracy score as: 0.80088

Random Forest Classifier

Select Data Analysis Option from Sidebar

Execute ML Operation

Click for Predictions

Training dataset retrieved

The predictions are based on Random Forest Classifier ML model .

Accuracy Score:

0.9993949127852687

ROC_AUC Score:

0.9999973475397598

By using this model we get the accuracy score as: 0.998

8. PREDICTION

Clearly, the best accuracy score is given by Random Forest Classifier which is 0.9976

Also the roc_auc score is : 0.9999

PG-DBDA Capstone Project

Project Guide: Mrs.Saruti Gupta

Developed by: Anokha Vinay Tigga and
Mayank Agrwal

Data Analysis

☐ none

☒ Charts

The predictions are based on Random Forest Classifier ML model .

Accuracy Score:

0.9993949127852687

ROC_AUC Score:

0.9999973475397598

9. CONCLUSION

Human activity recognition using a smartphone based system can be accomplished for both able-bodied and stroke populations; however, an increase in activity classification complexity leads to a decrease in HAR performance with a stroke population.

HAR predicts what percentage of activity user will perform and this can be used for deciding activities to be performed based on users health condition. The study results can be used to guide smartphone HAR system development for populations with differing movement characteristics.

10. FUTURE SCOPE

Since computer vision is a trending topic in these days, systems like Human Activity Recognition systems is quite useful and effective for solving a variety of application, whether it would be surveillance or monitoring, or aiding the elderly and blind people etc. This not only provide additional comfort to the end-users but also can be deployed into different Organizations in order to reduce the employ workload.

Human Activity Recognition system are of great importance in modern days due to the convenience and problems which the system offers and solves. Need of Activity recognition for monitoring and surveillance, video segmentation etc is of growing demand in which this system can greatly help.

This system can be incorporated in mobiles apps to further aid the elderly and blind people. It is cost-effective and an immense time saving system which is also prone to human errors. This system acts as a base solution for many other applications involving activity recognition. Hence, this system is very beneficial for both individual and organizations for general or specialize purposes.

11. BIBLIOGRAPHY

<https://databricks.com/glossary/pyspark>

<https://www.computerscijournal.org/vol13no23/human-activity-recognition-system/>

<https://towardsdatascience.com/streamlit-hands-on-from-zero-to-your-first-awesome-web-app-2c28f9f4e214>

<https://archive.ics.uci.edu/ml/index.php>

<https://archive.ics.uci.edu/ml/datasets/Heterogeneity+Activity+Recognition>