

Final Report

Instructor Device Activity Detection for VedinKaksa App - Mini Research Project

M.Madhumitha

14th June, 2019


Objective and Problem Statement

Vedinkaksa was built to enable and demonstrate the next-generation of smart-classrooms which is real time-feedback driven so that the instructor is always notified of the attention and involvement of the student during the lecture. Based on this real time feedback, the instructor can evaluate himself and change his/her teaching pattern to make the class more engaging as well as monitor the activity of the students in the classroom.

Modus-Operandi

In the Vedinkaksa classroom, every student possesses a smartphone/tablet which runs on the Android OS (App is developed for Android).

Whereas the Instructor possesses two devices - one is a primary device whereas the other is a peripheral device.



Once the classroom session begins, everybody including the students as well as the teacher log onto the app to register their attendance for the session.

The teacher then opens a pdf document (assumption for the time being) which covers the course material and it appears for all the students and he/she begins to teach on the primary device. But it is not necessary that the teacher has to teach ONLY from the pdf document, it is also possible that he/she teaches on the board or administers the lecture verbally.

My goal was to detect whether the device used by the teacher to instruct the class is in use or not. I.e whether it is in active state or inactive/idle state.

Once the classroom session begins, the app starts monitoring the engagement of the student and his concentration levels. This report is sent to the




instructor realtime in periodic predetermined intervals.

Here, there arises two cases -

1. Either the teacher is using the device and teaching the students - Device is in ACTIVE state
or
2. The teacher is not using the device while teaching the students - Device is in IDLE state

If the device is in active state, then the notification of the activity of the students should go to the Peripheral Device and not the Primary Device since the normal teaching of the teacher must not be disrupted by the attention reports at any cost.

Else if the device is in the idle state, then the notification of the activity of the students should go to the Primary device as it won't be disrupting the teaching.



So in a nutshell, the task is to determine whether the notification should go to the primary device or the peripheral device on the teacher's side.

Steps of the Experiment

- 1.** Look at all the possible sensors present in the smartphone and identify which one would be useful to determine the device activity.
- 2.** Collect optimum amount of data from different devices and for equal amounts of time for all the devices to maintain unbiased.
- 3.** Filter the data, plot graphs and come up with measures to analyse the data and try to identify a clear pattern exhibited by the data.
- 4.** Feed the data to different Machine Learning pre-built frameworks and determine the prediction accuracy for testing on the training data as well as new data.

5. This state prediction should be fed back to the Vedinkaksa Server which accordingly routes the notifications.

Step 1 - Hypothesis

The activity state of the device can be determined by performing an analysis realtime of all the data-values given by the sensors inbuilt in the Android Device which will enable us to detect any form of motion/usage.

Typically, all Android Devices have the following sensors inbuilt into them :

Base Sensors

Accelerometer

Ambient Temperature

Magnetic Field Sensor

Gyroscope



Heart Rate (Latest Models only)

Light

Proximity

Pressure

Relative Humidity

But this is a generalization, the actual presence may vary from device to device depending upon the manufacturing brand as well as the model of the make.

Composite Sensor Types

<https://source.android.com/devices/sensors/sensor-types>

To identify the sensor values which will be useful in recognizing the device state, I went through multiple scholarly articles and established sources listed below:


<https://www.aaai.org/Papers/IAAI/2005/IAAI05-013.pdf>

<https://www.sciencedirect.com/science/article/pii/S1877050914008643>

Given below is a detailed justification for the same:

Sensors in Smartphones & their Effectiveness in Determining the State of the User

Sensor	Android 4.0 (API Level 14)	Android 2.3 (API Level 9)	Android 2.2 (API Level 8)	Android 1.5 (API Level 3)
TYPE_ACCELEROMETER	Yes	Yes	Yes	Yes
TYPE_AMBIENT_TEMPERATURE	Yes	n/a	n/a	n/a
TYPE_GRAVITY	Yes	Yes	n/a	n/a
TYPE_GYROSCOPE	Yes	Yes	n/a ¹	n/a ¹
TYPE_LIGHT	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a
TYPE_MAGNETIC_FIELD	Yes	Yes	Yes	Yes
TYPE_ORIENTATION	Yes ²	Yes ²	Yes ²	Yes
TYPE_PRESSURE	Yes	Yes	n/a ¹	n/a ¹
TYPE_PROXIMITY	Yes	Yes	Yes	Yes
TYPE_RELATIVE_HUMIDITY	Yes	n/a	n/a	n/a
TYPE_ROTATION_VECTOR	Yes	Yes	n/a	n/a
TYPE_TEMPERATURE	Yes ²	Yes	Yes	Yes




Given above are the sensors which are supported by Android Studio, and can be used to get information about the user's interaction with the phone/tablet, provided that they are present on the device.

The following are the devices present on most of the android devices although there are exceptions, and some models which don't possess some of these sensors.

1.Accelerometer (Hardware Sensor)

An accelerometer detects acceleration, vibration, and tilt to determine movement and exact orientation along the three axes. Apps use this smartphone sensor to determine whether your phone is in portrait or landscape orientation.

It can also tell if your phone screen is facing upward or downward. The accelerometer can also detect how fast your phone is moving in any linear direction.



Therefore, it can ultimately help determine the handling behavior of the user – especially if the user is moving/walking while holding the device.

2. Gyroscope (Hardware Sensor)

Gyroscope also provides orientation details and direction like up/down and left/right but with greater precision like how much the device is tilted. This is where it differs from accelerometer — gyroscope can measure rotation too but the former cannot.

So it can tell how much a smartphone has been rotated and in which direction.

Therefore, it is even useful in determining if the user is moving the device around in his hand.

3. Magnetometer (Hardware Sensor)

Works like a compass- determines which direction the device is held at. The absolute values of the magnetometer will not be useful to us, but rather the change in the value of the magnetometer over a time interval eg. 20 seconds in my experiment will be useful to determine the movement/handling behavior of the user.

4. Rotation Vector Sensor (Software Sensor)

ROTATION_VECTOR sensor represents 'virtual' sensor which combines data from different sensors (usually ACCELEROMETER and GYROSCOPE) and does some smart calculations to provide more accurate data rather than using raw data from ACCEL and GEOMAGNETIC_FIELD sensors. It basically represents an accurate fusion of the results from all the three sensors to obtain orientation information.

5. Touch Sensor (Hardware Sensor)

It is used to trigger any activity based on active inputs received by the user. It is the most important parameter in detecting activity by the user. The no. of touches by the user in a certain time interval (20 secs in my experiment) is taken into account and recorded in the csv file.

6. Typing

It is a derivative of the touch sensor – it can directly denote the activity of the user.


Other Sensors Possibly Useful for the Experiment but not able to Use:

6. Mic – Basically used to record audio in phones. There is no existing framework in android studio which enables the programmer to directly extract values from this sensor. So facing some challenge in the programming portion. But this sensor is also not that useful in determining the state/behavior of the instructor in the classroom because it is possible that the instructor might speak while the interacting/not interacting with the device. So, the necessity for its values in the application is ambiguous.

Other Sensors Not Useful for the Experiment:

GPS – Position on the Earth as determined by a satellite (Used in navigation systems)

Proximity Sensor- A proximity sensor makes use of an infrared LED and IR light detector to find out how close the phone is to an outside object. It used while making calls and when the phone is held to the face to make or receive a call, the sensor detects it and disables the touchscreen display to avoid unintended



input through the skin. It detects the object only upto a distance of maximum 10cm, so it wouldn't be useful.

Ambient Light Sensor-

The light sensor detects the lighting levels in the vicinity to adjust the display brightness accordingly. It is used in Automatic Brightness Adjuster to decrease or increase the brightness of the smartphone screen based on the availability of light.


Fingerprint Sensor-

Fingerprint sensor enables biometric verification to secure many smartphones today. It is a capacitive scanner that records your fingerprint electrically.

Barometer, Temperature & Air-Humidity Sensors -

It is useful in detecting weather changes and altitude.

Thus, I used linear accelerometer, gyroscope, magnetometer and Touch Sensor. - Base Sensors.



I have also captured the rotation vector values because it gives the culminated effect of accelerometer, gyroscope and magnetometer values and thereby gives the rotation acceleration.

Architecture

For doing this, I built an Application on Android Studio which takes the device permissions and captures the values into a csv file which keeps getting updated as the sensor values are obtained in realtime from the app when the device is switched on, the Vedinkaksa app is opened, and the teacher starts the Classroom session.

The activity of the student starts getting recorded and the notifications start appearing to the teacher in real-time, at periodic intervals set by the App developer.


The task is to decide the route of the notification - whether it is to the primary device or the peripheral device. This will be determined by a machine learning algorithm which will be running continuously on the Vedinkaksa server on the data which is sent from the device of the teacher.

Step 2 - Controlled Experiment Performed

After the application is developed, I had to collect the data from unbiased sources to come up with the activity state.

I enumerated that there are six possible methods of handling of the device for the instructor:

1. The Device is on the desk and ACTIVE
2. The Device is on the desk and INACTIVE
3. The Device is in the hands of the instructor and ACTIVE
4. The Device is in the hands of the instructor and INACTIVE

- 
5. The Device is in the hands of the instructor and he/she is walking (pacing back and forth) while ACTIVE
 6. The Device is in the hands of the instructor and he/she is walking (pacing back and forth) while INACTIVE

I collected data (sensor values) from 10 devices for the same and the models are listed below:

Asus Nexus 7

Sony Xperia Z3

Htc Phone

Samsung SM-T331

Samsung SM-T231

Samsung SM-T131

Vivo V3

LeEco Le 2



Poco Phone F1

Sony Xperia G3226

Method of Collection of Data

I collected data from each and every device for all the six possible configurations, equally, for homogeneous composition of data and to capture different behaviors of the device.

For the purpose of collecting the training data, I performed a controlled experiment by asking different participants to explain content from any pdf of their choice, and it was loaded into the app.

I timed their explanation and each configuration data was collected for a period of 5 minutes for each device. The sensor values keep changing continuously, but I collected the values for every 20 seconds and stored it in a CSV format in the phone's



internal memory. This is to be continuously streamed to the server.

So, obtained data :

8 devices * 5 minutes / 20 seconds * 6 configurations
 $= 8 * 5 * 60/20 * 6 = 8*5*3*6=$

4 hours of data from different devices.

Participants of Experiment :

I tried to collect data from people who were not fully aware of the objective and use case of my data so as to minimize biasing. I just informed them what they were supposed to do within the given time frame and told them how they were supposed to handle the device during the explanation, so as to obtain data for that particular configuration.

There were 9 participants and their characteristics are as given below :

1. (Summer Intern - Age 21 years, Female)
- 2.(Summer Intern - Age 18 years, Female)
- 3.(Summer Intern - Age 20 years, Female)
- 4.(Summer Intern - Age 20 years, Male)
- 5.(Summer Intern - Age 20 years, Male)
- 6.(PhD Scholar - Age 35 years, Male)
- 7.(Professor - Age 48 years, Female)
- 8.(Project Manager - Age 50 years, Male)
- 9.(PhD Scholar, Teaching Assistant - Age 23 years, Male)

While collecting the data, I manually observed them and noted down the label of the data (ACTIVE / INACTIVE) and filled it in the excel file after the experiment.

The columns are :

Acc x, Acc y, Acc z,

Gyr x, Gyr y, Gyr z,

Mag x, Mag y, Mag z,

Rot x, Rot y, Rot z,

Touch.

Step 3 - Analyzing & Interpreting the Data

The collected data cannot be used as such and needs to be processed.

The absolute difference between the consecutive values is calculated and labelled as

Abdif x, Abdif y, Abdif z (For Linear Accelerometer)

Abgyr x, Abgyr y, Abgyr z (For Gyroscope)

Abmag x, Abmag y, Abmag z (For magnetometer)

Abrot x, Abrot y, Abrot z (For composite sensor - Rotation Vector Sensor)

The touch value is intact - It measures how many times the user touched the screen in 20 seconds.

After this, the x,y and z components have to be combined into a single value. The Euclidean Distance Measure is used to quantify the difference between consecutive values.

```
acc = sum(power(abdif x,2),power(abdif y,2),power(abdif z,2))
```

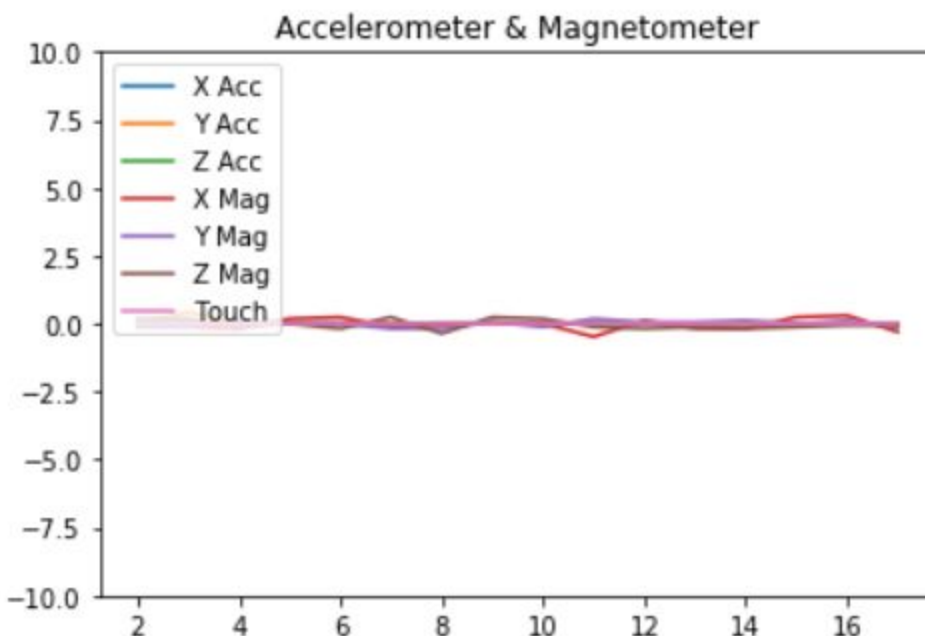
```
mag = sum(power(abmag x,2),power(abmag y,2),power(abmag z,2))
```

```
rot = sum(power(abgyr x,2),power(abgyr y,2),power(abgyr z,2))
```

The values are then plotted against in Jupyter Notebook using Matplotlib. A few screenshots are attached below : The detailed report can be found in the .ipynb file i have attached with the report.

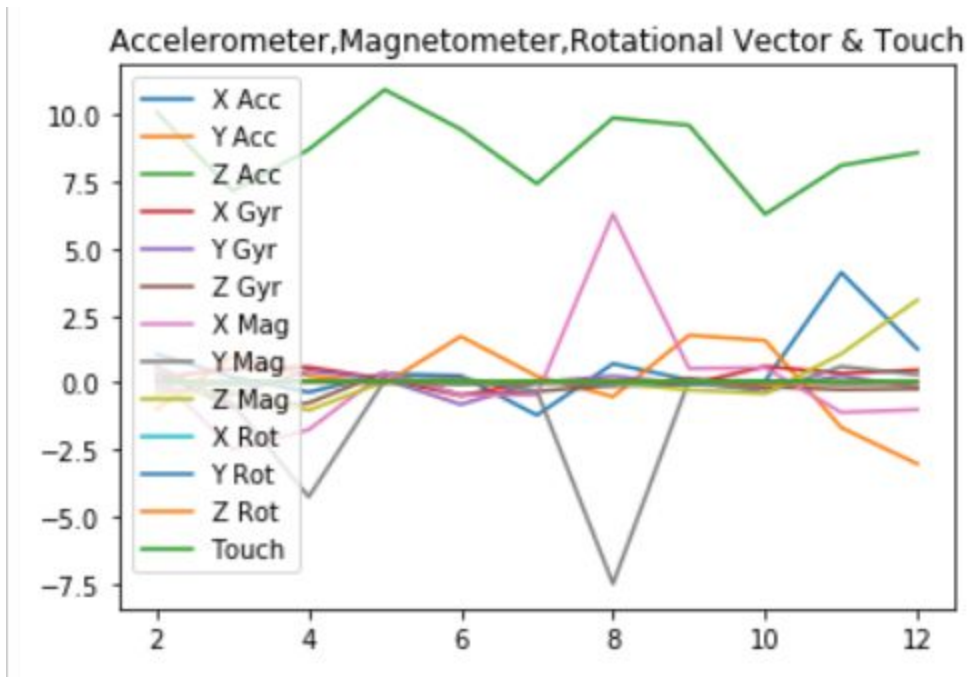
Graph Forms with sample graphs :

Idle - Ondesk



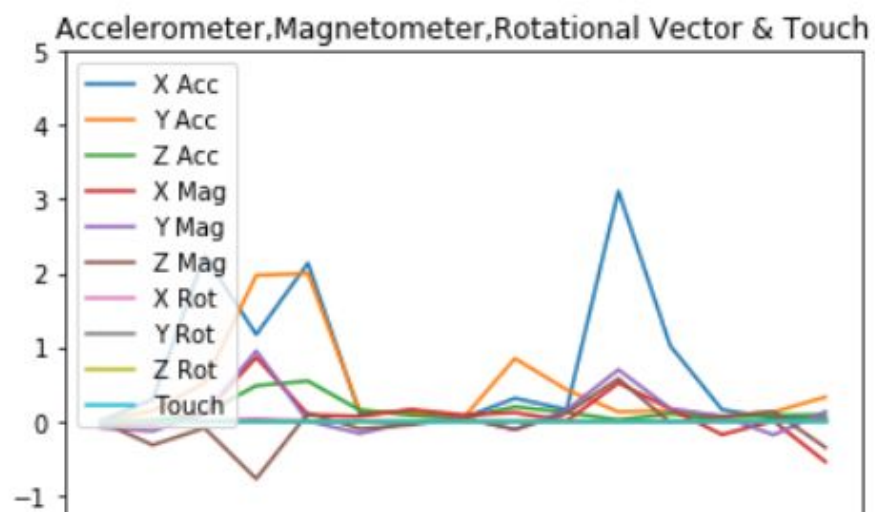
Observation : Very less deviation from the zero axis indicating no change in values and zero touch.

Idle - Moving



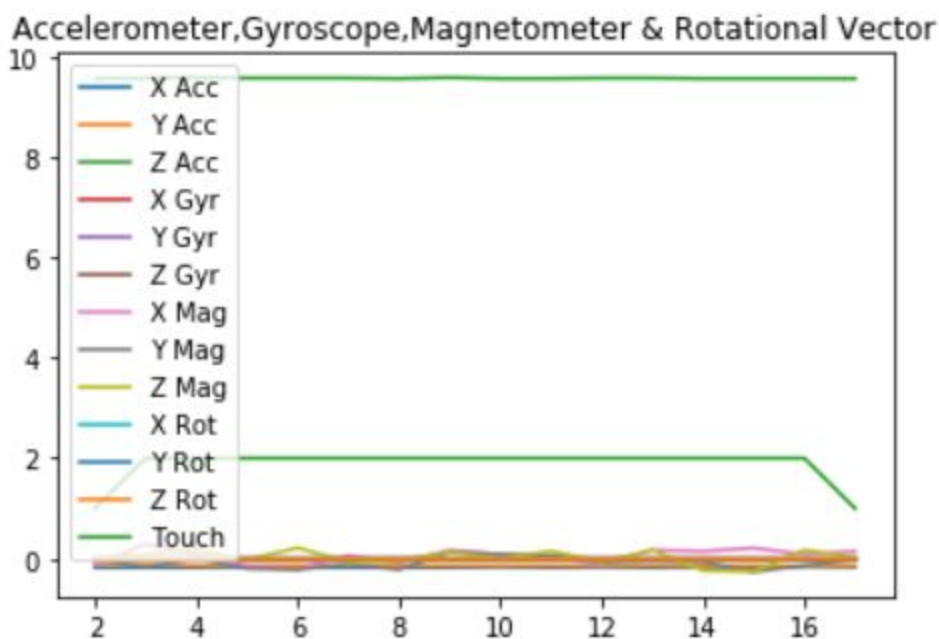
Observation : Significant deviation from the zero axis indicating change in values and no touch.

Idle - in hand



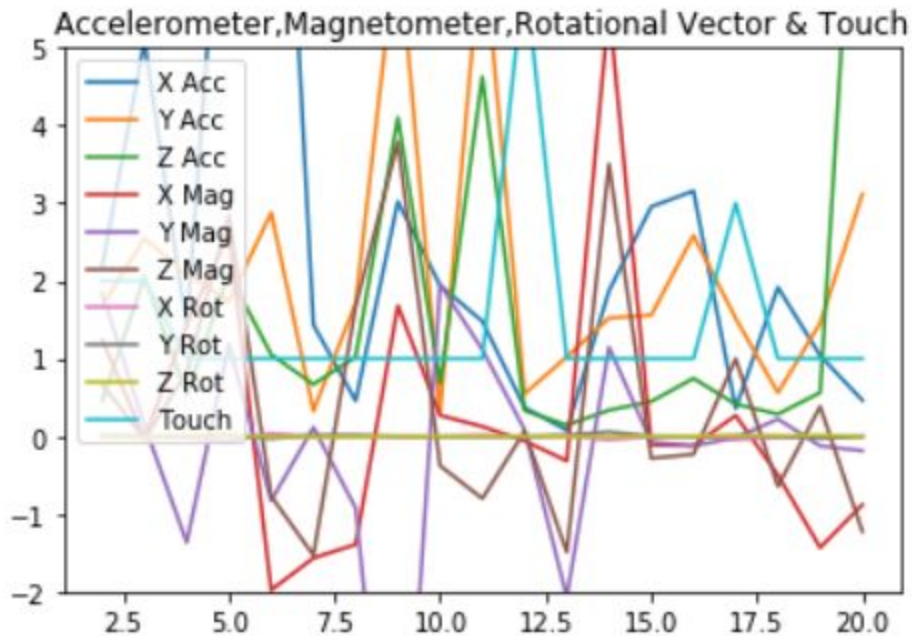
Observation : Significant deviation from the zero axis indicating change in values and no touch, and no manually observable pattern except for zero touch throughout.

Active- on Desk



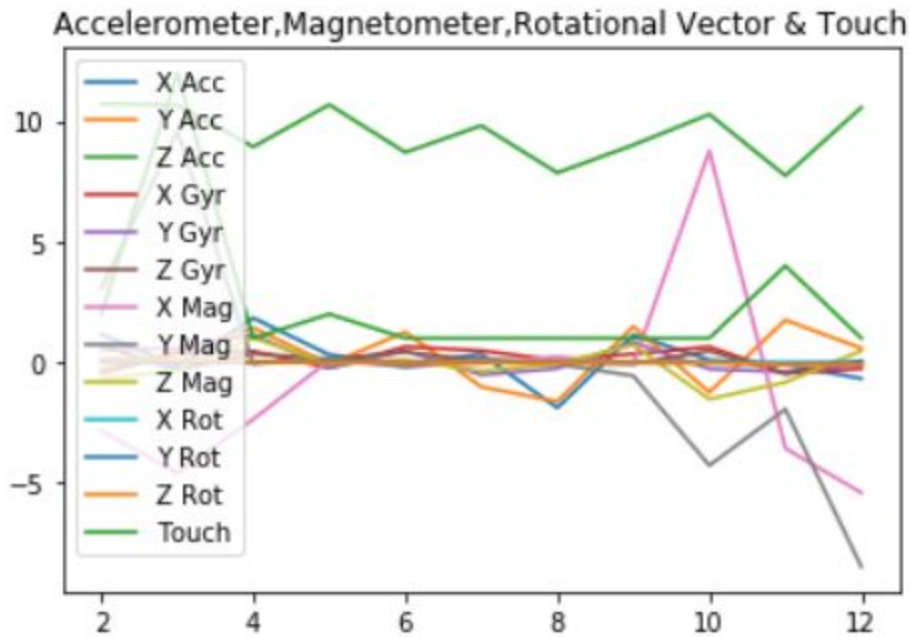
Observation : Very little deviation from the zero axis indicating change in values and no touch, and no manually observable pattern except for non-zero touch throughout.

Active - In Hand



Observation : Significant deviation from the zero axis indicating change in values and no touch, and no manually observable pattern except for non-zero touch throughout.

Active - Moving



Observation : Significant deviation from the zero axis indicating change in values and no touch, and no manually observable pattern except for non-zero touch throughout.

Overall Observation :

For the collected data, no useful information can be obtained from the gyroscope values and the magnetometer values. (and therefore even Rotational Vector Values)

So, for my case study, only (Linear Accelerometer, Touch) values and (Touch) values are to be tested.

Step 4 - Feeding the data to Machine Learning Algorithm

I used Sci-kit Library and imported the Logistic Regression Framework.

Feeding only the Touch column gave a prediction accuracy of **0.9905660377358491 (99%)** when tested on some random sampled, labeled data which was not used for training.

Feeding the (Touch, Linear Accelerometer) columns gave a prediction accuracy of **0.9581749049429658 (96%)** when tested on some random sampled, labeled data which was not used for training.




Conclusion

Thereby, from the above results, it can be concluded that touch alone will suffice and in fact, even yield better results when compared to using both the Linear accelerometer as well as touch. This is very contradicting to the intuitive course of thought, but it is the result.

But there are definitely limitations and very uncommon exceptions which can occur and result in misprediction by the system.

Limitations & Possible Exceptions

If the user continuously keeps reading from the device without touching it for a long duration of time, then the state is actually ACTIVE but will be mispredicted as inactive.



I tried to artificially create this scenario, but was unable to get a zero touch situation for the pdf files, as the participants tended to perform some form of interaction with the device screen like zooming in & out ,swiping,scrolling. But this might be possible in a device with a huge screen.

Also, before generalizing the result, and not just for the app, more data must be collected and more parameters and machine learning algorithms have to be implemented & computed.

Future Scope

If increasing the complexity of the system is not a problem, then the system can be made more rigorous by using Computer Vision and identifying the face orientation of the user to predict if he's looking into the device, or looking away which implies that he is inactive. This can be used along with the touch.

—

—

—

—