MOTIVATION

- The humongous use of smartphones have come on the back of the extreme vulnerability of sensitive data which on being lost or stolen can be misused at various levels.
- Inefficiency of present authentication techniques in ensuring real-time security.
- Primary motive of ensuring greater security and lessening privacy breach by developing a learning model to implement a response to unknown touch.
- Overcoming Smudge Attack and Shoulder Surfing.
- A more comprehensive security measure than the existing ones.

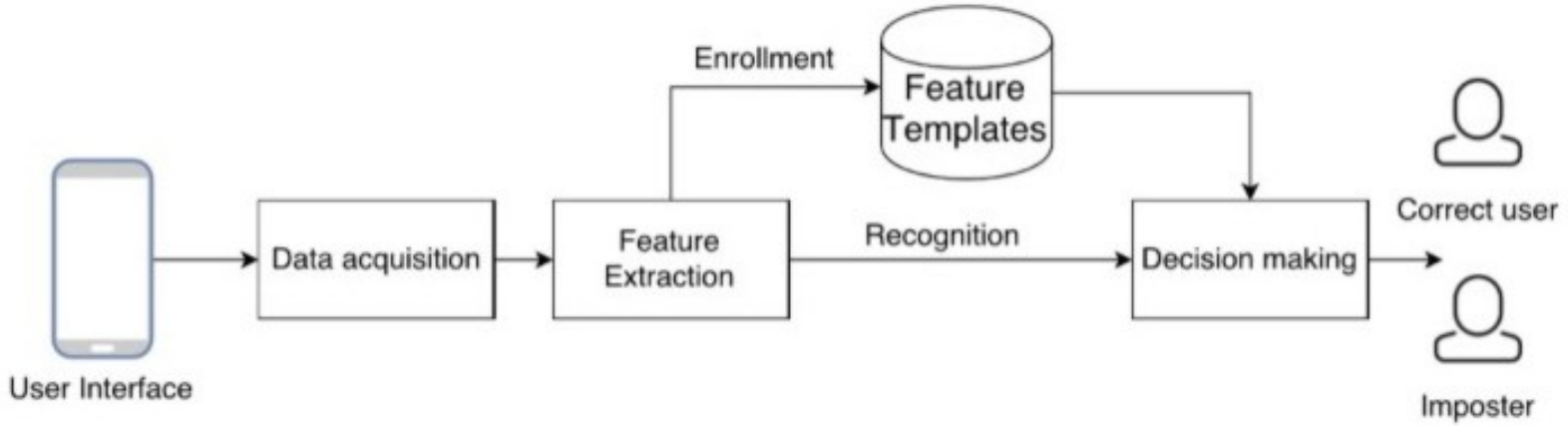
Resources used

- Android Studio, a development platform for android applications, was used for building the user interface to collect the touch-related data.
- Jupyter Notebook, an open source web application, was used to implement the learning algorithms in Python 3.

OBJECTIVE

- The primary objective remains securing mobile devices using unknown touch behaviour.
- Creating a user interface to collect the touch data in form of Android Application.
- Selection parameters such as size of touch including the length and the width of the ellipse formed on touching, pressure exerted and the time duration of the touch and subsequent lift operation.
- Implementing a learning model to train and test the data generated and then predict if the user is genuine or otherwise.

PROPOSED SOLUTION



Flow chart of our approach

Designing User Interface:

To collect data and perform experiment, an Android application was developed however it can be developed for any other mobile platfrom with multi-touch support also. The application guides the user through creating training data and store it in the CSV format, which can be shared later.

- **Data Acquisition.**

Android platform provides an API to interact with Touch Input in form of Java class “Motion Event”, an object of this class is created for each touch event.

long	getDownTime() Returns the time (in ms) when the user originally pressed down to start a stream of position events.
long	getEventTime**()** Retrieve the time this event occurred, in the SystemClock.uptimeMillis() time base.
float	getPressure**()** getPressure(int) for the first pointer index (may be an arbitrary pointer identifier).
float	getSize**(int pointerIndex)** Returns a scaled value of the approximate size for the given pointer *index* (use getPointerId(int) to find the pointer identifier for this index).
float	getTouchMajor**(int pointerIndex)** Returns the length of the major axis of an ellipse that describes the touch area at the point of contact for the given pointer *index* (use getPointerId(int) to find the pointer identifier for this index).
float	getTouchMinor**()** getTouchMinor(int) for the first pointer index (may be an arbitrary pointer identifier).

Description of the selected features

Features	Descrpition	Importance
EventTime	It provides the total duration of the Touch Event	Since users agility,taping speed varies, so different users will have different patterns of eventTime
DownTime	Provides the duration for which the finger is pressed down.	Similar to EventTime it will also provide unique pattern for each user
Pressure	The amount of pressure applied by the user on the Touch device.	Important behaviour characteristics as every user applies varied amount of pressure.
TouchSize	Every Touch generates an ellipse and TouchSize provides area of ellipse in terms of pixels	Each user will have varried shape and size of there Finger.
MajorAxis	Size of MajorAxis of the generated ellipse.	Each user will create an ellipse of varried dimension and thus MajorAxis and MinorAxis will make an unique pattern
MinorAxis	Size of MajorAxis of the generated ellipse.	MajorAxis and MinorAxis will make an unique pattern

Training and Decision Making

For training and decision making first we had to select the learning model which will perform best on our select features, for that we implemented some of the popular Machine Learning Algorithms and plotted their Accuracy on bar graph as shown below:

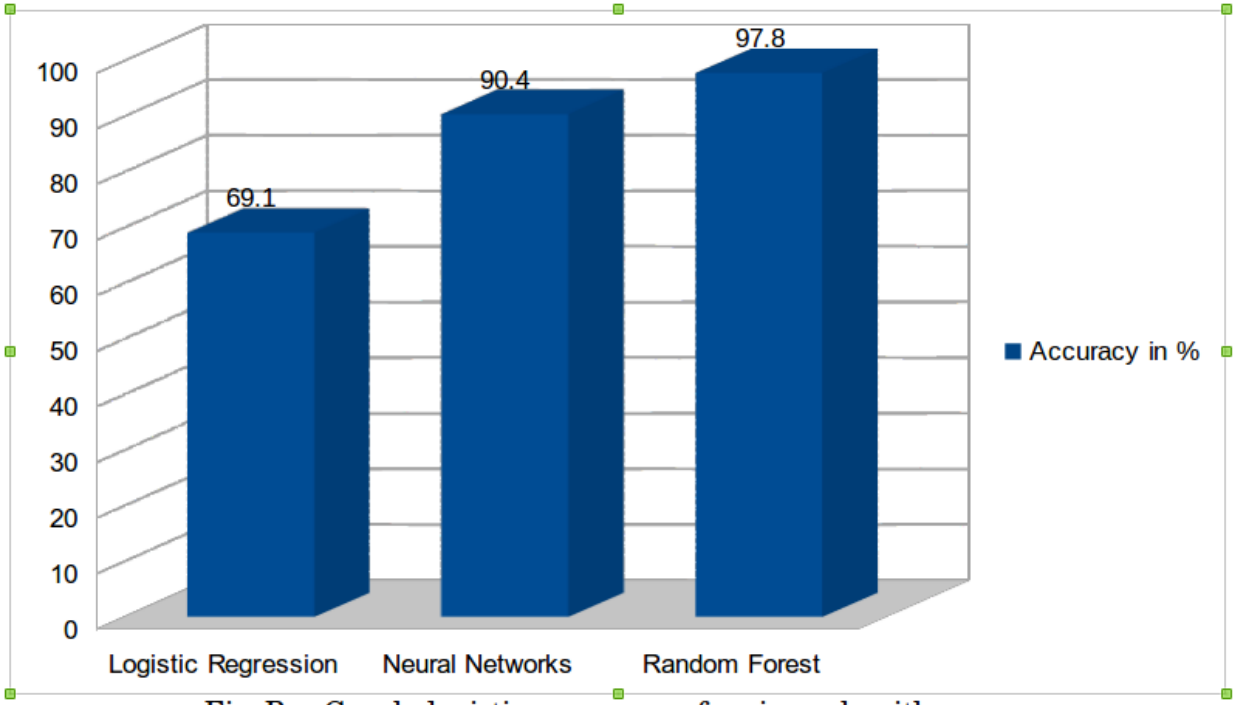


Fig. Bar Graph depicting accuracy of various algorithms.

Training and Decision Making

After observing the accuracy graph the Random Forest gives highest accuracy on our features set..

We implemented the Random forest in Python using Pandas and sklearn libraries.

For implementation the collected data-set was added with the touch data of other users and after that it was labeled properly with values 1 (for genuine user) and 0.

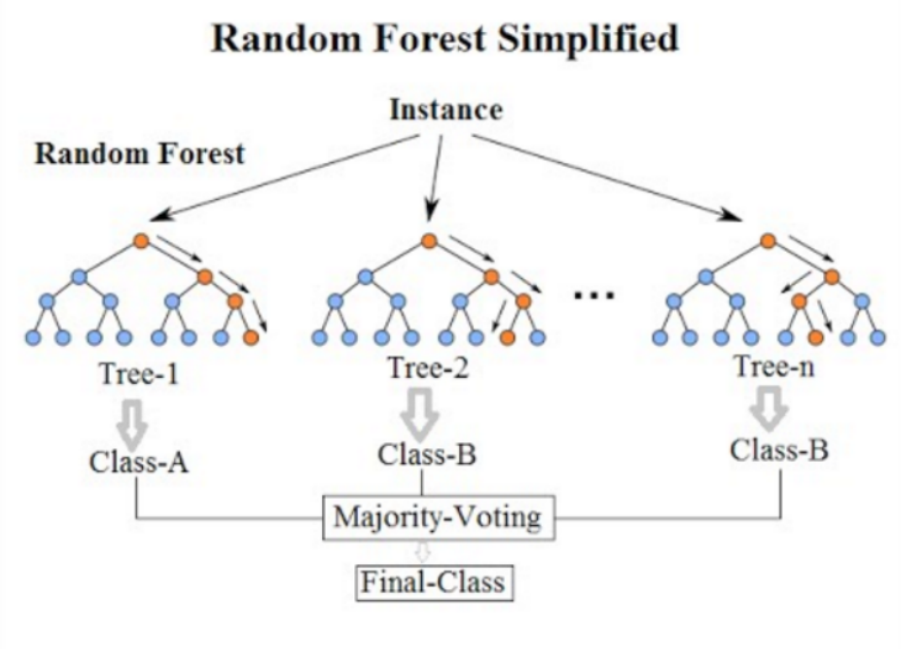
The resulting CSV file was then converted to vector form using the pandas library.

The vectorized data was then fed to the Random Forest Model.

After the training, the model will be used to make decision on the basis of the touch features collected in real time to detect the user is genuine or not.

Random Forest

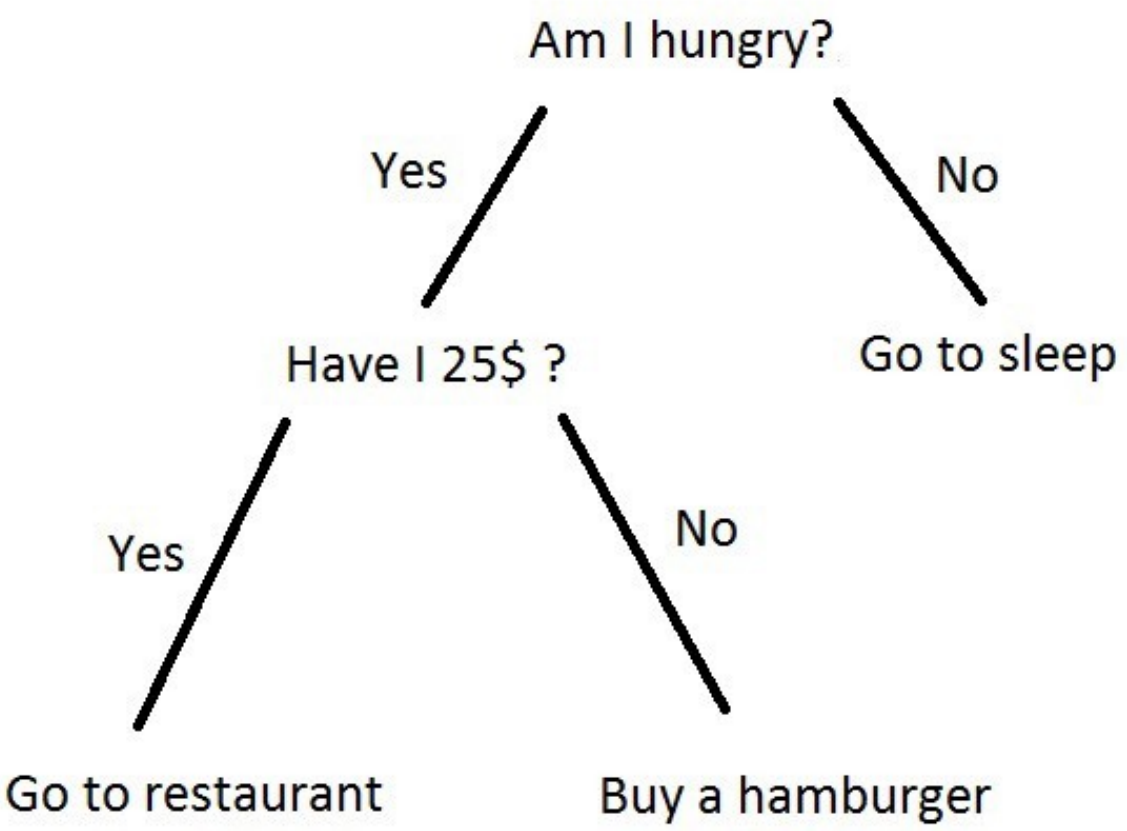
A random forest is a data construct applied to machine learning that develops large numbers of random decision trees analyzing sets of variables. This type of algorithm helps to enhance the ways that technologies analyze complex data.



Understanding a Decision Tree

- A decision tree is the building block of a random forest.
- We can think of decision trees as a flowchart of questions asked about our data, eventually leading to a predicted class (or continuous value in the case of regression).
- This is an interpretable model because it makes decisions how we do in real life: we ask a series of questions about the data until we eventually have arrived at a decision.

Understanding a Decision Tree



Terms Related to Decision Tree

Impurity

Impurity is when we have a traces of one class division into other. This can arise due to following reason.

1.We run out of available features to divide the class upon.

2.We tolerate some percentage of impurity (we stop further division) for faster performance. This threshold % impurity is called Gini Impurity.

Entropy

Entropy is degree of randomness of elements or in other words it is ***measure of impurity.***

Mathematically, it can be calculated with the help of probability of the items as:

$$H = -\sum p(x) \log p(x)$$

Information Gain

Suppose we divide the classes into multiple branches as follows, the information gain at any node is defined as:

Information Gain (n) = Entropy(x) — ([weighted average] *entropy (children for feature))

Dividing efficiently based on maximum information gain is key to decision tree classifier.

EXPERIMENTAL RESULTS

- In the approach followed by us and in the algorithms used, we implemented a basic model of Random Forest Classifier using Python 3 to build the classification model.
- The model was trained with 70% of the data and the remaining 30% data was used as testing set.
- The overall accuracy obtained was about 98%.
- Logistic Regression implemented on the same dataset gave an accuracy of about 68%, hence performing rather poorly.

CONCLUSION

- Inferring from the above result its rather clear and evident that our algorithm is efficient enough to discern one user from the other using the touch behaviour of a user.
- Two major aspects of our implementation were independent of each other.
- This drawback shall be overcome in further progress.
- Using a server based mechanism to feed the data and obtain result.
- Another approach includes implementing a simpler and more efficient algorithm that runs natively on the smartphone.

Performance Analysis of Similar Works

- Gait Recognition
- It observes the walking behaviour of a user through hardwares.
- Keystroke Dynamics
- Observes the typing behaviour and frequency as features.
- Gesture based Authentication
- Users are asked to perform certain fixed gestures whilst authenticating the smartphone.
- Accuracy obtained was about 77%.
- A related implementation that used multiple finger touch data and employed SVM gave an overall accuracy of about 95%.