Biometric Liveness Detector

CSE 535: Mobile Computing

Rahul Murali
Arizona State University
Tempe, United States
rmurali7@asu.edu

Sruthi Sathyamoorthy
Arizona State University
Tempe, United States.
ssathy10@asu.edu

Bhagyasri Musunuru
Arizona State University
Tempe, United States
mbhagyasri@asu.edu

Vineethkrishna Vemireddy
Arizona State University
Tempe, United States
vvemired@asu.edu

***Abstract*—Spoofing attacks are a serious threat to the biometric systems and its utility. Smart Phone applications are used as a solution across many practical problems. In this project, we present an intelligent android application, to detect the biometric liveness and anomaly detection. Machine learning models such as Random Forest, Support Vector Machine (SVM), Hierarchial Clustering (agglomerative clustering), Logistic Regression, Multi-Layer Perceptron Classifier (MLP), Adaboost classifiers were used to test the veracity of the signal. The models were evaluated using accuracy, F1 scores,False Accept Rate, False Reject Rate, Half total error, etc. A timing analysis for training and testing the models were performed and the aggregated results were displayed.**

***Index Terms*—mobile application, biometric liveness detection, machine learning.**

I. Introduction

Brain signal investigation finds its application across multiple domain such as entertainment, rehabilitation and most importantly to treat brain diseases. The study of brain diseases such as Alzheimer's, epilepsy, schizophrenia, and stroke involves investigating the brain signals. Brain signals exhibit some peculiarity which is not shared by other biometrics. This makes them unique. Having said that, there are many challenges that needs to be addressed while investigating the brain signals. One such challenge associated with the liveness detection is the spoofing attack. In this project, we present an intelligent mobile application that can detect if the single is live or not using machine learning models. For this project we have choosen, Random Forest, Support Vector Machine, Hierarchial Clustering, Logistic Regression, Multi-Layer Perceptron Classifier, Adaboost classifiers.As we know that different Machine Learning models will produce different accuracies, we take into account six different models and then display the results from the best model to the user. This ensures high accuracy.

II. Project Setup and Permissions

The project setup and architecture consists of two systems.

1) Android device:
   a) Device model : Pixel 4a.
   b) Android version: Android 11.
2) Fog Server :
   a) Device model : HP X360.
   b) OS : Windows 10.
   c) Intel i7 10th Gen.
   d) RAM : 16 GB.

Identify applicable funding agency here. If none, delete this.

III. Architecture/Pipeline diagram

Figure 1 represents the architectural or pipeline diagram of the project.

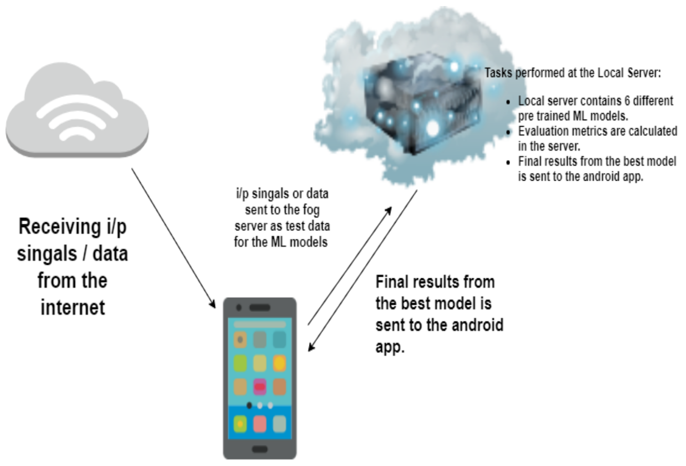

Tasks performed at the Local Server:
Local server contains 6 different pre trained ML models.
Evaluation metrics are calculated in the server.
Final results from the best model is sent to the android app.
Receiving i/p singals / data from the internet
i/p singals or data sent to the fog server as test data for the ML models
Final results from the best model is sent to the android app.

Figure 1. Example of a figure caption.

IV. Server Setup

1) The server uses a Flask RESTful webservice and is hosted on the local server.
2) Pretrainied Machine learning models such as Random Forest, Support Vector Machine, Hierarchial Clustering, Logistic Regression, Multi-Layer Perceptron Classifier, XGBoost are running on the server.
3) The server identifies the type of the input (single line or multi-line input) then sends the test data accodingly for the prediction to the model.
4) Various model evaluation metrics is calculated in the server and the results from the best model are sent to the android app.

V. Flow Chart

Figure 2 represents the Flowchart describing the overall procedure.

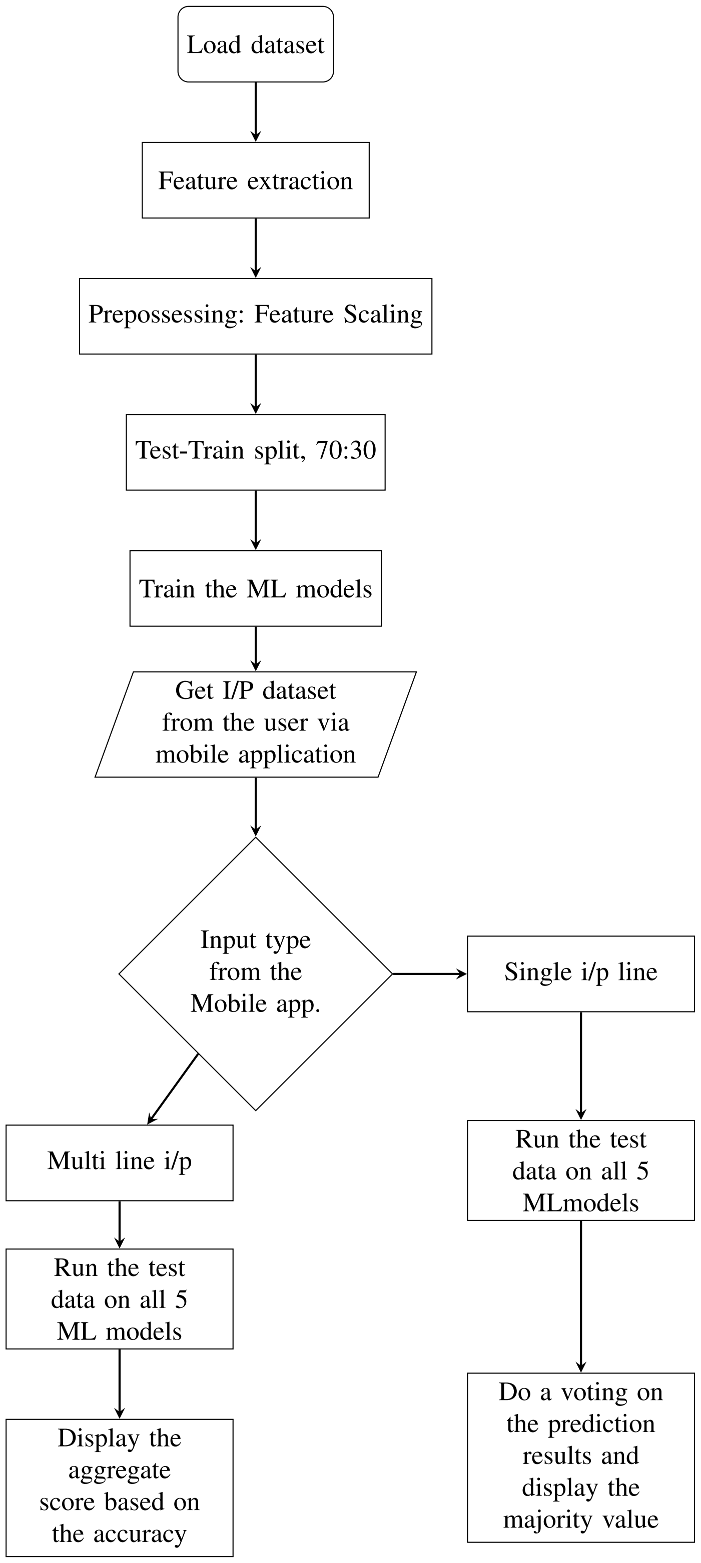

Load dataset
Feature extraction
Prepossessing: Feature Scaling
Test-Train split, 70:30
Train the ML models
Get I/P dataset from the user via mobile application
Input type from the Mobile app.
Single i/p line
Multi line i/p
Run the test data on all 5 MLmodels
Run the test data on all 5 ML models
Display the aggregate score based on the accuracy
Do a voting on the prediction results and display the majority value

Figure 2. Flowchart describing the overall procedure

VI. Implementation

1) Interactive UI to receive the input brain signals and to display the prediction results to the user: The app can receive and send data through internet. User must enter IP address and Port number for the app to connect to the machine. This enables receiving data to the app and sending data to the server. We have implemented this using Client server approach.

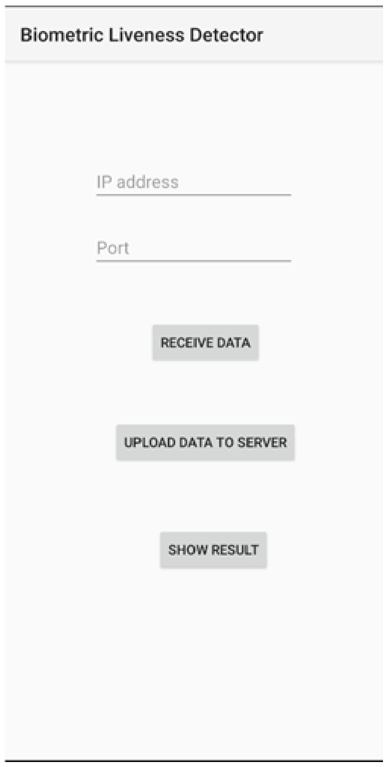
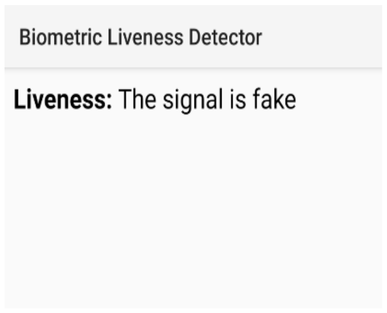


Figure 3. Biometric Liveness Detector Mobile application

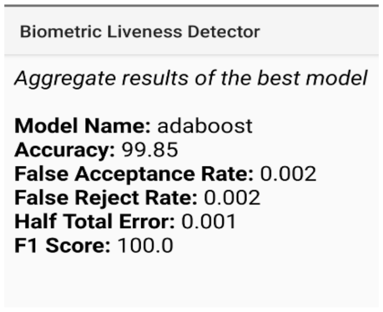
To receive data, user enters IP address and Port number and click on 'Receive Data'. The application connects to the server and receives data. On click of 'Upload Data to server', the app connects to the server as per the specified IP address and uploads received data to the server, where the models are trained, and aggregate results are computed.



Liveness: The signal is fake

Figure 4. Results for single input

The app is capable of receiving data in two modes, one is single input data and other is multiple input data. On click of


Biometric Liveness Detector

Aggregate results of the best model

Model Name: adaboost
Accuracy: 99.85
False Acceptance Rate: 0.002
False Reject Rate: 0.002
Half Total Error: 0.001
F1 Score: 100.0

Figure 5. Results for multiple input

'Show Result', the results are shown as per type of the input received. The results are shown in Fig 4 and Fig 5.

2) Fog Server Implementation: For this project, we have used the local server to deploy the machine learning models and to calculate the various model evaluation metrics.

The server uses a python Flask RESTful service and various methods according to the user's action.

- Action : Getting the input data from the server.
  Method: We obtain the brain signal data from the internet by using the GET method.
- Action : Uploading the input file with signals to the server.
  Method: We upload the input data to the server by using the POST method.
- Action : Server receiving the input file which is the test data to the machine learning models that is deployed in the server.

- Action : Model evaluation parameters are calculated in the server using python functions.

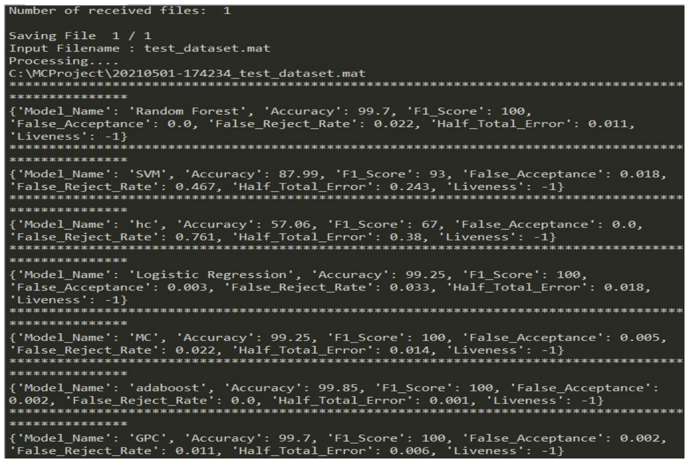

```
Number of received files:  1

Saving File  1 / 1
Input Filename : test_dataset.mat
Processing....
C:\MCProject\20210501-174234_test_dataset.mat
*********************************************************************************************
***************
{'Model_Name': 'Random Forest', 'Accuracy': 99.7, 'F1_Score': 100,
'False_Acceptance': 0.0, 'False_Reject_Rate': 0.022, 'Half_Total_Error': 0.011,
'Liveness': -1}
*********************************************************************************************
***************
{'Model_Name': 'SVM', 'Accuracy': 87.99, 'F1_Score': 93, 'False_Acceptance': 0.018,
'False_Reject_Rate': 0.467, 'Half_Total_Error': 0.243, 'Liveness': -1}
*********************************************************************************************
***************
{'Model_Name': 'hc', 'Accuracy': 57.06, 'F1_Score': 67, 'False_Acceptance': 0.0,
'False_Reject_Rate': 0.761, 'Half_Total_Error': 0.38, 'Liveness': -1}
*********************************************************************************************
***************
{'Model_Name': 'Logistic Regression', 'Accuracy': 99.25, 'F1_Score': 100,
'False_Acceptance': 0.003, 'False_Reject_Rate': 0.033, 'Half_Total_Error': 0.018,
'Liveness': -1}
*********************************************************************************************
***************
{'Model_Name': 'MC', 'Accuracy': 99.25, 'F1_Score': 100, 'False_Acceptance': 0.005,
'False_Reject_Rate': 0.022, 'Half_Total_Error': 0.014, 'Liveness': -1}
*********************************************************************************************
***************
{'Model_Name': 'adaboost', 'Accuracy': 99.85, 'F1_Score': 100, 'False_Acceptance':
0.002, 'False_Reject_Rate': 0.0, 'Half_Total_Error': 0.001, 'Liveness': -1}
*********************************************************************************************
***************
{'Model_Name': 'GPC', 'Accuracy': 99.7, 'F1_Score': 100, 'False_Acceptance': 0.002,
'False_Reject_Rate': 0.011, 'Half_Total_Error': 0.006, 'Liveness': -1}
```

Figure 6. Results displayed on the server for multi-line i/p.

- Action : The aggregate results are being sent to the android application.
  Method: The android application uses the GET method to receive the aggregate results.

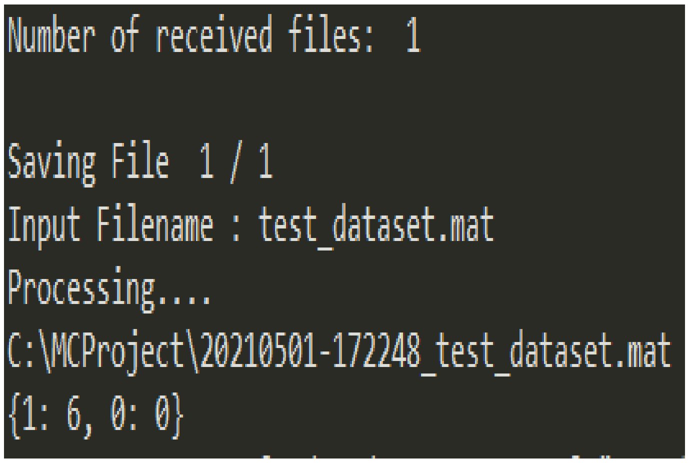

```
Number of received files:  1

Saving File  1 / 1
Input Filename : test_dataset.mat
Processing....
C:\MCProject\20210501-172248_test_dataset.mat
{1: 6, 0: 0}
```

Figure 7. Results displayed on the server for single-line i/p.

Images of the results displayed on the server for training data and testing data are shown in figure 6 and figure 7.

3) Training data: We have basically extracted the input training data from the raw data set that was given to us in the form of a .mat file.
So, originally we had 2226 samples(both the live and non-live data sets combined together), where the live signal had data points for 120 seconds and the non-live signal had data points for 30 seconds.So, in order to avoid the time confusion we had just considered the time frame for the initial 3 seconds(480 data points as the sampling rate is 160Hz) to make it easy for the feature extraction stage by not losing much of the information

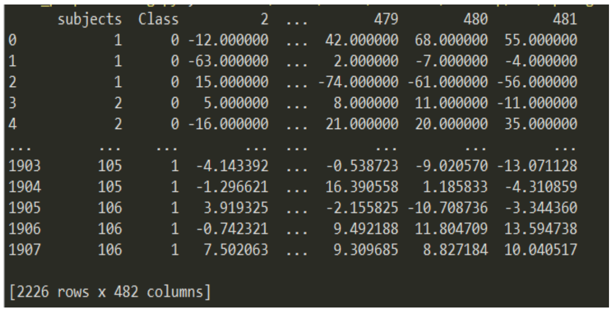
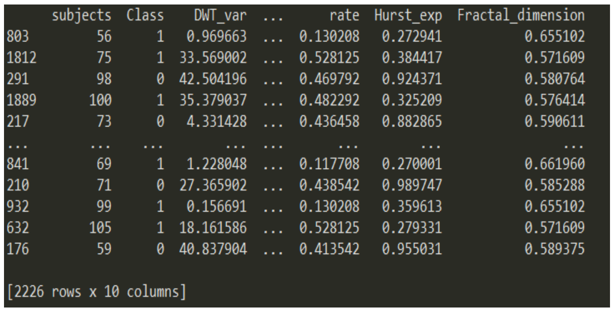

```
      subjects  Class          2  ...        479        480        481
0            1      0 -12.000000  ...  42.000000  68.000000  55.000000
1            1      0 -63.000000  ...   2.000000  -7.000000  -4.000000
2            1      0  15.000000  ... -74.000000 -61.000000 -56.000000
3            2      0   5.000000  ...   8.000000  11.000000 -11.000000
4            2      0 -16.000000  ...  21.000000  20.000000  35.000000
...        ...    ...        ...  ...        ...        ...        ...
1903       105      1  -4.143392  ...  -0.538723  -9.020570 -13.071128
1904       105      1  -1.296621  ...  16.390558   1.185833  -4.310859
1905       106      1   3.919325  ...  -2.155825 -10.708736  -3.344360
1906       106      1  -0.742321  ...   9.492188  11.804709  13.594738
1907       106      1   7.502063  ...   9.309685   8.827184  10.040517

[2226 rows x 482 columns]
```

Figure 8. Dataset

4) Feature Extraction Techniques: The following feature extraction techniques were used.

1) Fourier Transform (BETA Band)
2) Zero Crossing
3) Zero Crossing Rate
4) Discrete Wavelet Transform(DWT) - Variance
5) Discrete Wavelet Transform(DWT) - Skewness
6) Discrete Wavelet Transform(DWT) - Kurtosis
7) Petrosian Fractal Dimension
8) Hurst Exponent

5) Normalization: To normalize the data, standard scaling was used.



```
      subjects  Class     DWT_var  ...      rate  Hurst_exp  Fractal_dimension
803         56      1    0.969663  ...  0.130208   0.272941           0.655102
1812        75      1   33.569002  ...  0.528125   0.384417           0.571609
291         98      0   42.504196  ...  0.469792   0.924371           0.580764
1889       100      1   35.379037  ...  0.482292   0.325209           0.576414
217         73      0    4.331428  ...  0.436458   0.882865           0.590611
...        ...    ...         ...  ...       ...        ...                ...
841         69      1    1.228048  ...  0.117708   0.270001           0.661960
210         71      0   27.365902  ...  0.438542   0.989747           0.585288
932         99      1    0.156691  ...  0.130208   0.359613           0.655102
632        105      1   18.161586  ...  0.528125   0.279331           0.571609
176         59      0   40.837904  ...  0.413542   0.955031           0.589375

[2226 rows x 10 columns]
```

Figure 9. Features Extracted

6) Machine Learning Models: The following machine learning models were used to predict if the signal is live or not.

1) Random Forest
2) Support Vector Machine (SVM - Polynomial kernal)
3) Hierarchial Clustering - Unsupervised Learning
4) Logistic Regression
5) Multi-Layer Perceptron Classifier (MLP)
6) AdaBoost
7) Gaussian Process Classifier(GPC)

VII. Results

A. Execution time analysis

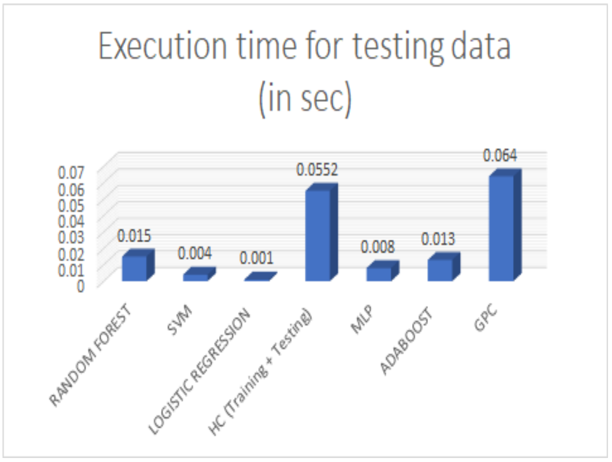


Figure 10. Execution time analysis across various ML models on the testing data

Table I
Training and Test times evaluation.

Model Name	Training time	Testing Time
Random Forest	0.1616	0.015
Support Vector Machine	0.016	0.004
Logistic Regression	0.0189	0.001
HC (training time + testing time)	0.0552	0.0552
Multi-Layer Perceptron Classifier	1.398	0.008
Adaboost classifier	0.1170	0.013
GPC	31.7456	0.064

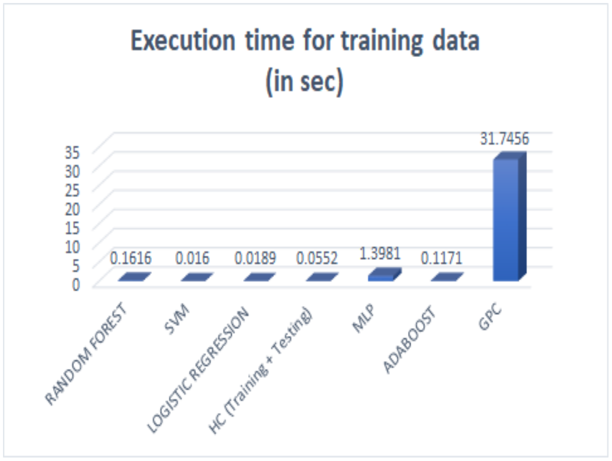


Figure 11. Execution time analysis across various ML models on the training data

B. Model Evaluation Metrics

1) Accuracy: Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

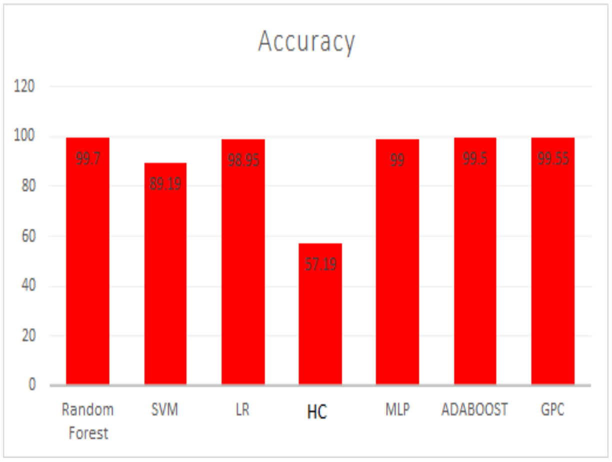


Figure 12. Accuracy scores of various ML models

2) F1 Score: F1-score: This is the harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases than the Accuracy Metric.

3) False Accept Rate: The false acceptance ratio (FAR) is a unit used to measure the average number of false acceptances within a biometric security system. It measures and evaluates the efficiency and accuracy of a biometric system by determining the rate at which unauthorized or illegitimate users or singals are verified on a particular system.

4) False Reject Rate: False Rejection Rate (FRR) is the percentage of identification instances in which authorised persons or signals are incorrectly rejected.

5) Half Total Error (HTER): Half Total Error is a fraction of $\frac{FAR + FRR}{2}$

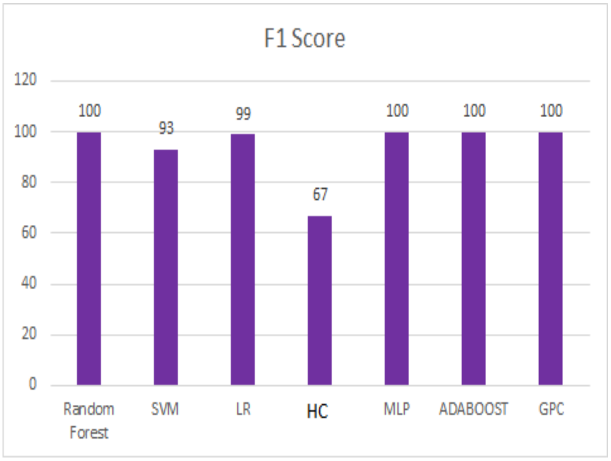


Figure 13. F1 of various ML models

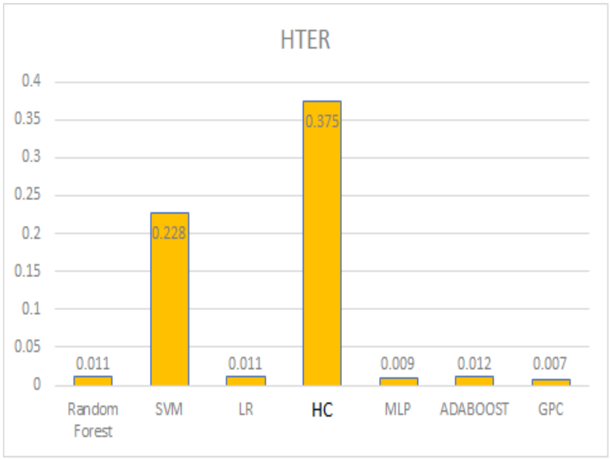


Figure 16. HTER of various ML models

C. Sample results

Table II
SAMPLE RESULTS

Model Name	Accuracy	F1 Scores	FAR	FRR	HTER
Random Forest	99.7	0.0	0.022	0.011	100
SVM	89.19	0.019	0.437	0.228	93
LR	98.95	0.01	0.012	0.011	99
HC	57.19	0.0	0.75	0.375	67
MLP	99	0.007	0.011	0.009	100
ADABOOST	99.55	0.002	0.022	0.012	100
GPC	99.55	0.003	0.011	0.007	100

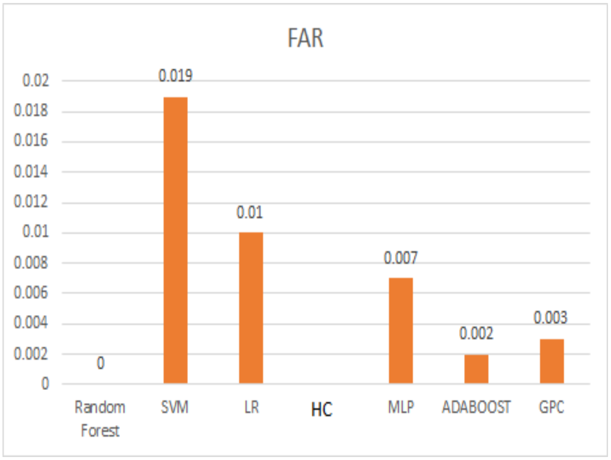


Figure 14. FAR of various ML models

Below link has excel sheet with the performance metrics for 7 models and 8 feature extraction techniques. CLICK HERE TO VIEW THE RESULTS.

VIII. Attack Vector Generation.

A. Additive White Gaussian noise (AWGN)

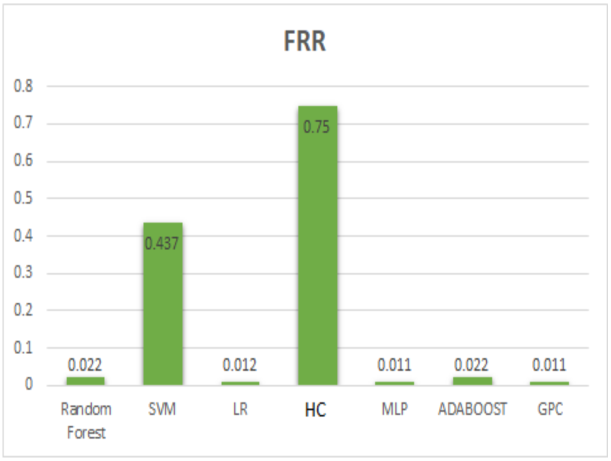


Figure 15. FRR of various ML models

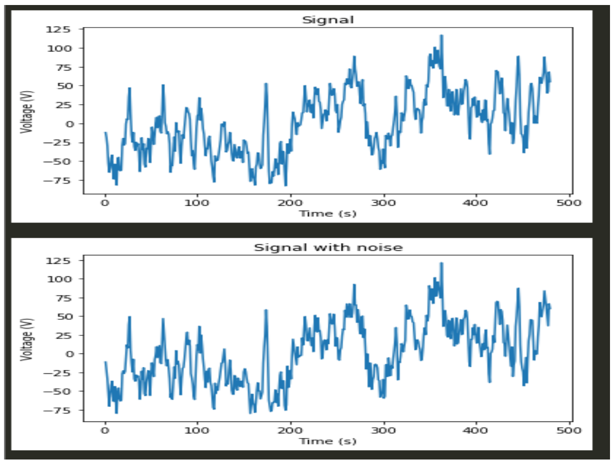


Figure 17. HTER of various ML models

Figure 12 shows the input signal and the signal with noise addition. Additive White Gaussian noise (AWGN) is mixed with the input signal and the results are displayed. This analysis is for one particular signal to visualize to signal in the pre and post noise stages. The attack signals are then generated based on the process [y(t) + noise], where y(t) is the input signal.

B. Generative adversarial networks model (GAN)

For the GAN model, we had fetched the feature vector as an input for the model, as the raw input data could be causing some issues for the model in the training phase [2].

Table III
TRAINING AND TEST TIMES EVALUATION FOR GAN MODEL.

Model Name	Training time (mins)	Testing Time (mins)
GAN model	92	45

1) Execution time analysis for GAN model:

IX. SUGGESTED METHOD TO FIND OUT OF THE GIVEN SIGNAL IS LIVE AND NOT PRE-RECORDED.

In order to check if the signal is not prerecorded, we propose a new feature that could be added to our mobile application. Electroencephalography is a process to record the electrical activity of the brain. Through electroencephalogram one can record the brain activity.

Electroencephalography is a process to record the electrical activity of the brain. EEG sensors [1] can be used to measure the electrical activity in the outer layer of the brain. During the electroencephalogram the EEG sensors are placed on the head of the participants and then the brain waves are detected by the electrodes non-invasively.

Within a single second, the EEG sensors can record up to several thousands of snapshots of electrical activity generated in the brain. These signals are sent to the amplifiers and then to the server. From there, the data can be directly sent to the pre trained machine learning model that is deployed on the server as a test data for evaluation. This is one way to make sure that the results are live and not pre recorded. In addition to that we could also add a time stamp to the input signal to know when the signal was recorded.

X. CONCLUSION

Biometric systems can be decieved easily. Therefore, the application of biometric liveness detection is very crutial and should be implemented in systems that has higher security risks.
In this project, we developed mobile application that acts as a verification server for liveness detection for brain signals. Data will be sent to app on the phone through Internet, the app will analyze it using relevant machine learning techniques such as Random Forest, Support Vector Machine (SVM), Hierarchial Clustering (agglomerative clustering), Logistic Regression, Multi-Layer Perceptron Classifier (MLP), Adaboost classifiers were used and the results were delivered (live or not live). The project successfully focused on the detecting a live input that is not artificially generated. The results from the machine learning model was evaluated based on accuracy, F1 scores, False Accept Rate, False Reject Rate, Half total error. A timing analysis of the testing and training of the model was made.

XI. ACKNOWLEDGMENT

We would like to express our special gratitude to Prof. Ayan Banerjee for giving us this golden opportunity to work on the project on detecting the veracity of the brain signal by using a mobile application and Mohammad Javad for guiding us through each phase of the project. It was a wonderful learning opportunity that helped us hone our skills in android, deep learning, server implementation.

REFERENCES

[1] Pore, Madhurima and Sadeghi, Koosha and Chakati, Vinaya and Banerjee, Ayan and Gupta, Sandeep K.S., Enabling Real-Time Collaborative Brain-Mobile Interactive Applications on Volunteer Mobile Devices, 2015,Association for Computing Machinery, New York, NY, USA, 46–50,5, Paris, France, HotWireless '15

[2] EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals,Kay Gregor Hartmann and Robin Tibor Schirrmeister and Tonio Ball,2018,1806.01875,arXiv,eess.SP