Distinguish abnormal individuals

from

Neck laser data

The City College of New York

DSE I9800: Capstone Project

Supervised by: Professor Jie Wei

Submitted By: Md Ayub Ali Sarker

Fall 2020

Introduction

Biomedical signal processing is one the most popular field in Digital Signal Processing. This involves analysis of signal measurements to provide useful information that can used to make clinical decisions. Engineers are discovering a new way to process these signals using a variety of mathematical formulae and algorithms. In this project work we are going to apply biomedicals signal processing technique on neck laser data, with machine learning to get remote sense of indvidual's health conditions.

Neck Laser data contains the human pulse vibrations over the neck artery collected by a Laser Doppler Vibrometer. Data file in .mat format. Each .mat file is the 1D signal values in the time spectrum like waves, electricity, mechanical vibrations etc.

Neck laser original data is in time domain. In order to process this signal files, we first converted to frequency domain by Fast Fourier Transform (FFT). Then from signal in frequency domain we first derived heartbeat of each individual by two developed methods and took the heartbeat from the method with lowest error. Then we extracted peak frequencies in ranges 0~7hz, 2.6~10Hz, 11~20Hz, 21~30Hz, 31~40Hz and 41~50Hz from the signal in frequency domain, make each one as feature of an observation.

We have extracted some features from original signal in time domain. Those are zero crossing rate average, spectral rolloff average, spectral centroid average, spectral bandwidth average, poly features average, rms average and spectral flatness average.

After extracting Heartbeat and other features on all the files we have, we extracted the level mapping the ground truth we have. Although we have total of 39 observations but in ground truth file, we have 19 observations that matched the observation we have. So, Total of 19 observations we have to apply machine learning.

As we know small data may have some issues in feeding model. Models may suffer from overfitting and high bias. So, we used TCGAN [3] to generate some synthetic data. Later used that synthetic data together with original data we feed our model.

We developed four classification models. Those are AdaBoost, Decision Tree, KNN and Random Forest. We feed each model and with original data and synthetic plus original data separately. We saw that Adaboost (67% on original data and 71% on synthetic plus original data) is good choice.

Problem statement

Neck laser data is collected from three group of peoples. Group A are age between 18~30, Group B are age between 31~50 and Group C are age between 50+. Each observation contains multiple signal files. Each data files are 1D signal values in time domains. We also have participant information that contains pulse, health conditions of each individual. Note that we don't have level information for all the observation we have. We have only 19 observations that have level.

Extracting features from signal in time and frequency domain, we determined heartbeat of each individual and heath condition (Normal and Not Normal) using signal processing and machine learning technique.

Motivation

Our laser sensor is remote sensor. It can measure bio sign from 10 meterd. In theory, it could be 400~500 meters. This can be used to determine remote bio sign. Like covid-19 and other serious illness. That's why I am motivated.

Neck Laser Data

Neck Laser data is a biomedical data set. This data set contains the human pulse vibrations over the neck artery collected by a Laser Doppler Vibrometer [5]. Each single set of data is collected for a person from multiple left-side and right-side scans and saved in .mat format. Each .mat file is the 1D signal values in the time spectrum like waves, electricity, mechanical vibrations etc. We have total of 235 files mat files of 39 persons of three groups (A: 18~30, B:31~50 and C:50+). Each file is index by subID and each file data are the 60 seconds with sampling rate 44,100 Hz. Here is the one of the signal files looks like.

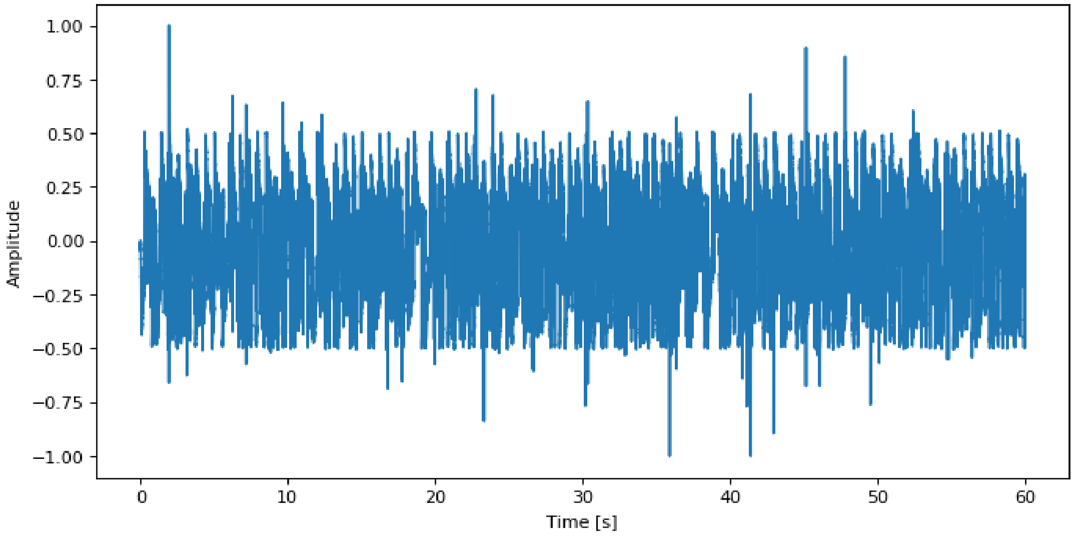


Figure-1: Original signal in time spectrum

Together with neck laser data that we have a participant data that contains information about participant who participate in collecting neck laser data which can be index by subId. In Participant data, we have information like Health issue, Pulse, Age, Sex, Blood pressure, Ethnicity and Weight. In this project work we used only Health issue and Pulse. We used health issue to determine heatlh condition (Normal or Not Normal) and Pulse for validating our developed heartbeat extraction method.

Laser Doppler vibrometer

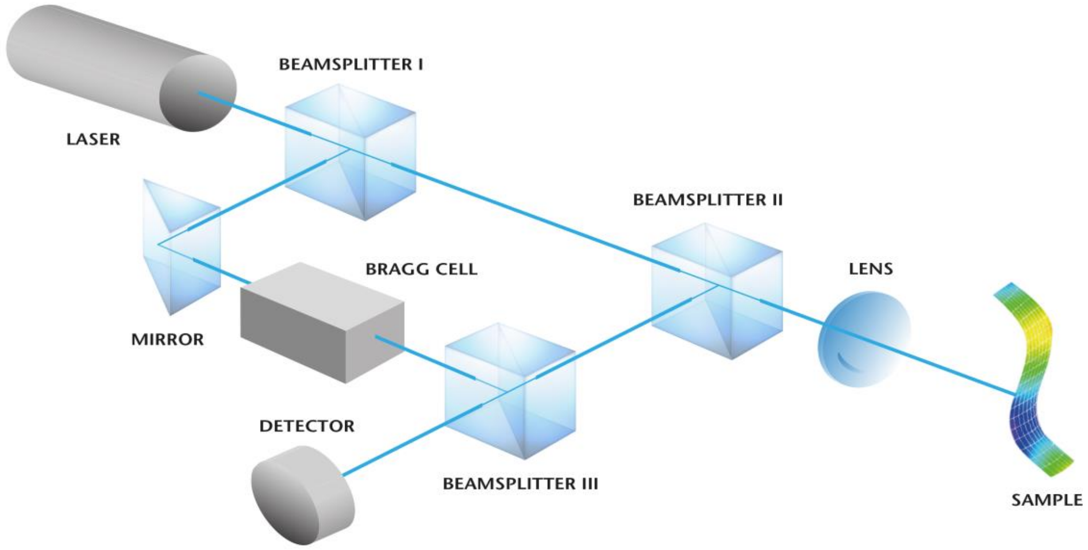


Figure-2: Laser Doppler vibrometer

The beam of a laser is split by a beam splitter (BS 1) into a reference beam and a measurement beam. After passing through a second beam splitter (BS 2), the measurement beam is focused onto the sample, which reflects it. This reflected beam is now deflected downwards by BS 2 (see figure) and is then merged with the reference beam onto the detector. Depending on velocity and displacement back scatted light changed in frequency and phase. Characteristics of motions/vibration are completely containing in back scattered light. The superimpose of back scatter light and reference beam create modulated output signal revealing droplet shift in frequency.

Solution to the problem

In order the address the problem mentioned in the problem statements, we did the following major tasks for each individual. Those are

- Extract Heartbeat
- Extract features in time and frequency domain
- Extract Level from participant data
- Feature Selection

- Feed the model with original data
- Generate 200 synthetic data using TCGAN [3]
- Feed the model with original plus synthetic data

Heartbeat Extraction

We developed two procedures to extract heartbeat from signals. We first converted original signal in time spectrum to frequency spectrum using Fast Fourier Transform. The we removed trend from the signal. Then we identified several local maxima in Fourier magnitudes for each files of an observation in range 0.8~2.5Hz. Then we apply our procedure to derived heartbeat of an individual.

- **Clustering Approach**: We took five top peaks from signal in frequency domain for each signal file of an individuals. Then we multiply each peak frequency with 60 to get heartbeat. Later we did Kmean1D [6] clustering of all peak heartbeat and took cluster center with maximum neighbors as the heartbeat for that individual. Here are the steps we followed in deriving heartbeat in this approach.

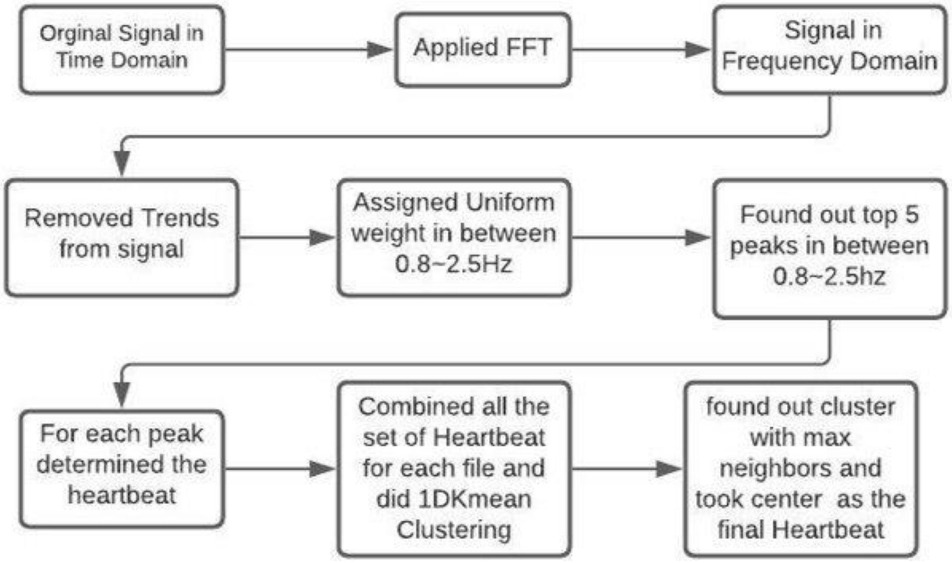

Orginal Signal in Time Domain
Applied FFT
Signal in Frequency Domain
Removed Trends from signal
Assigned Uniform weight in between 0.8~2.5Hz
Found out top 5 peaks in between 0.8~2.5hz
For each peak determined the heartbeat
Combined all the set of Heartbeat for each file and did 1DKmean Clustering
found out cluster with max neighbors and took center as the final Heartbeat

Figure-3a: Heartbeat extraction clustering approach steps

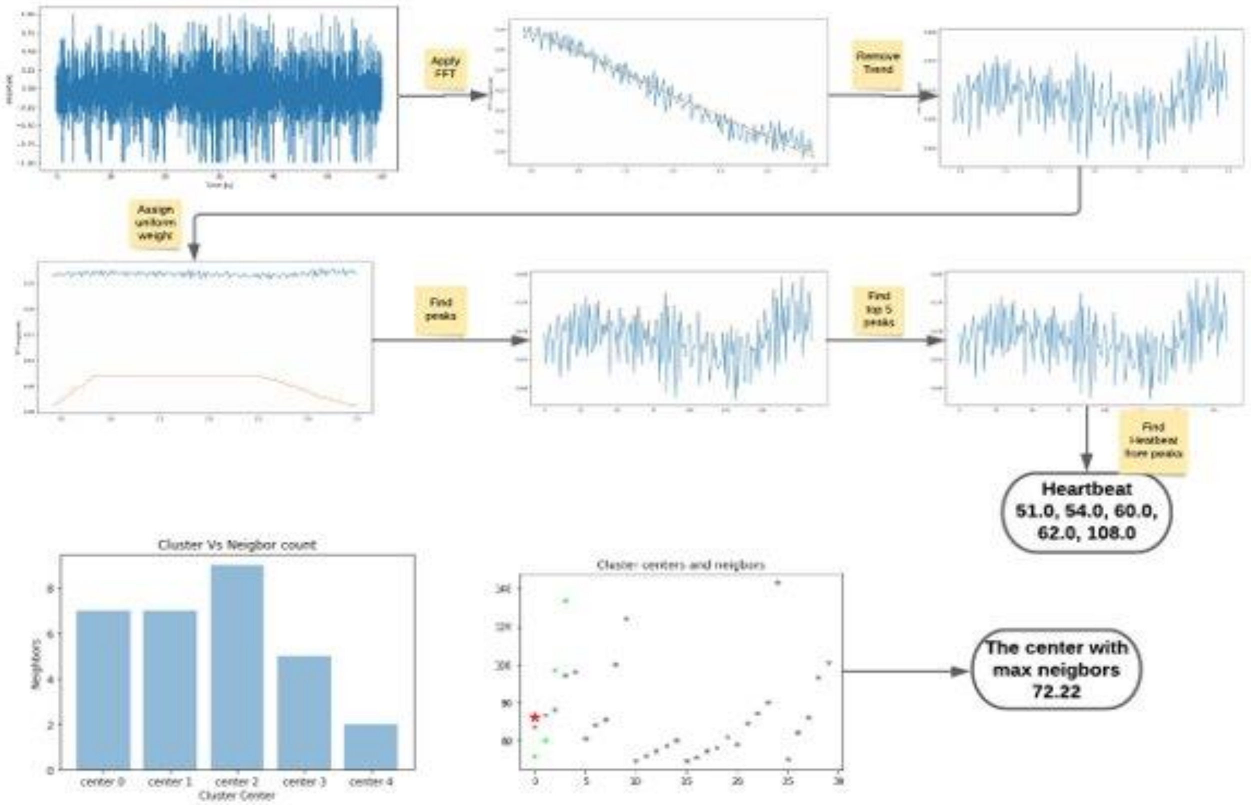


Figure-3b: Heartbeat extraction clustering approach

- **Average Approach**: Like Clustering approach. Found out top five peak frequencies and calculated heartbeat from peak frequencies. Then averaged all the peak heartbeat from all peak's heartbeat from all the files and took averaged heartbeat as the final heartbeat of individuals.

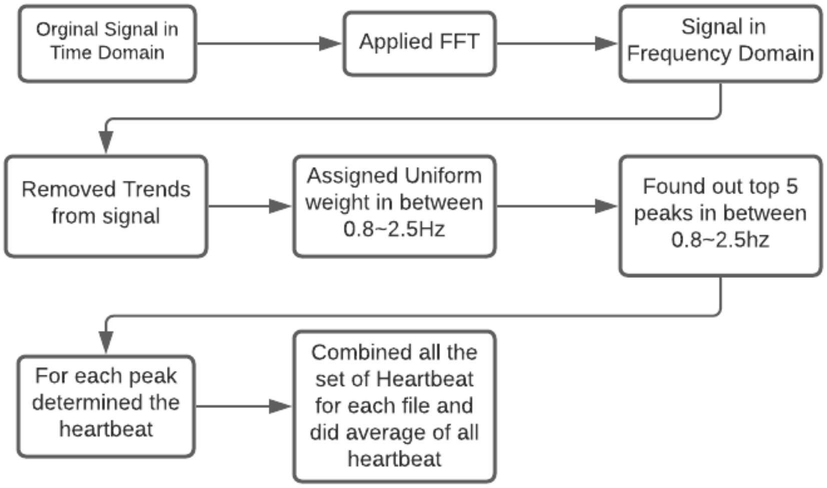

Orginal Signal in Time Domain
Applied FFT
Signal in Frequency Domain
Removed Trends from signal
Assigned Uniform weight in between 0.8~2.5Hz
Found out top 5 peaks in between 0.8~2.5hz
For each peak determined the heartbeat
Combined all the set of Heartbeat for each file and did average of all heartbeat

Figure-4a: Heartbeat extraction average approach steps

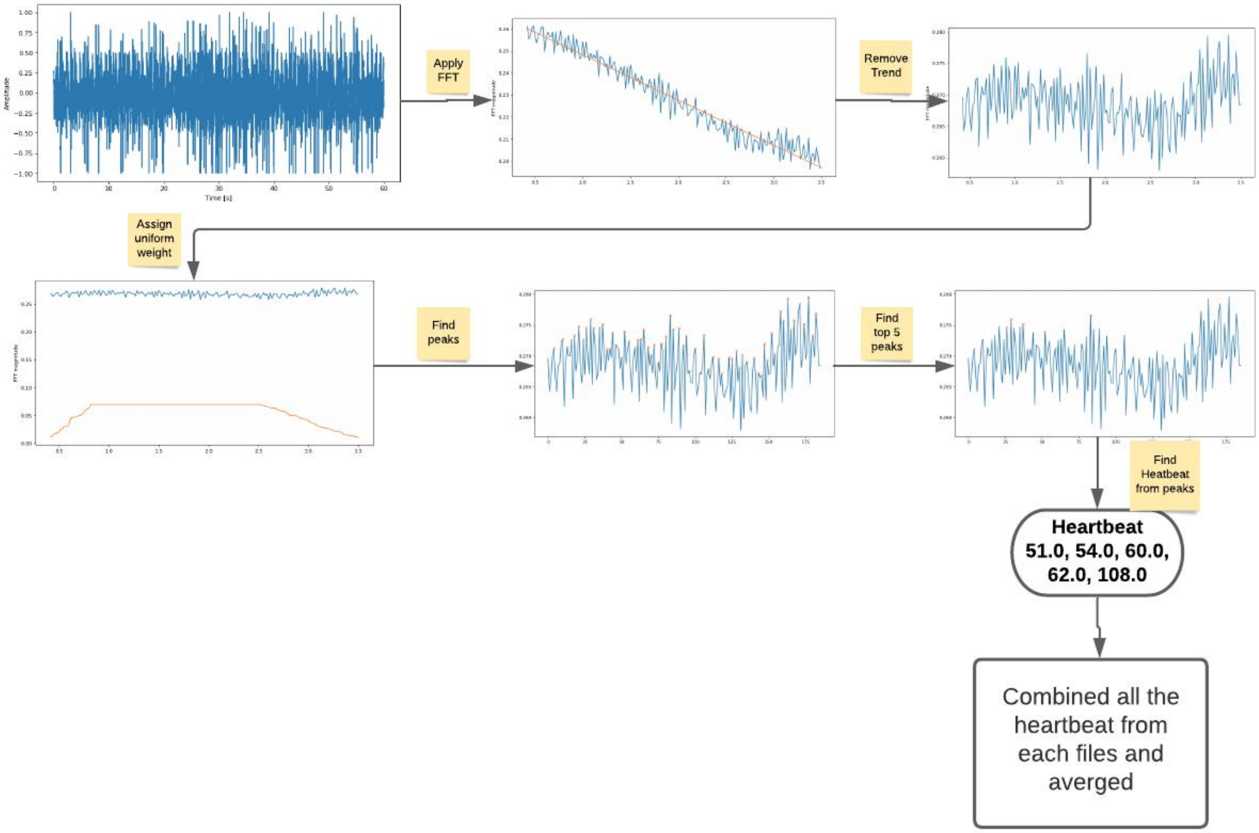


Figure-4b: Heartbeat extraction average approach

After computing heartbeat using both approaches, we compared the heartbeat with actual heartbeat from the participation data (Pulse in Participant data). We calculated error of each approach. Here the error

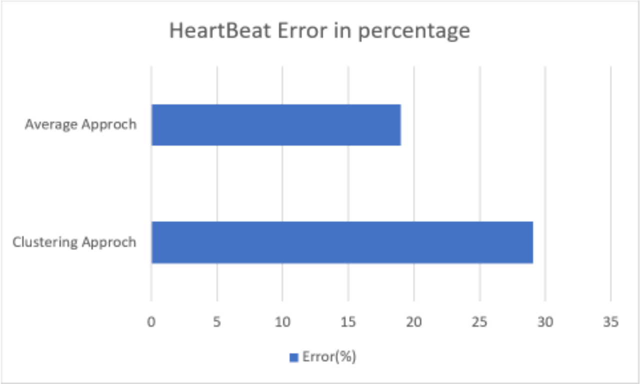


Figure-5: Error in Heartbeat Calculation

We can clear see that Average approach (19%) is better than Clustering Approach (29.09%). So, we took heartbeat calculated from average approach as the final heartbeat to use in later machine learning approach.

Extract features in time and frequency domain

One the other important task of this works is to extract features that plays a vital role in feed machine learning algorithm. We divided feature extraction subsection. One is feature extraction in time domain and other is feature extraction frequency domain.

- **Feature extraction in frequency spectrum:** We divided the signal in frequency domain between 0~50Hz into following into 0~0.7Hz, 0.8~2.5hz, 2.6~10hz, 11~20Hz, 21~30Hz, 31~40Hz and 41~50Hz band and found out the peak frequency in each band, using clustering approach describe in heartbeat section and defined that peak frequency as the feature in that band. Here are the steps we followed in this process.

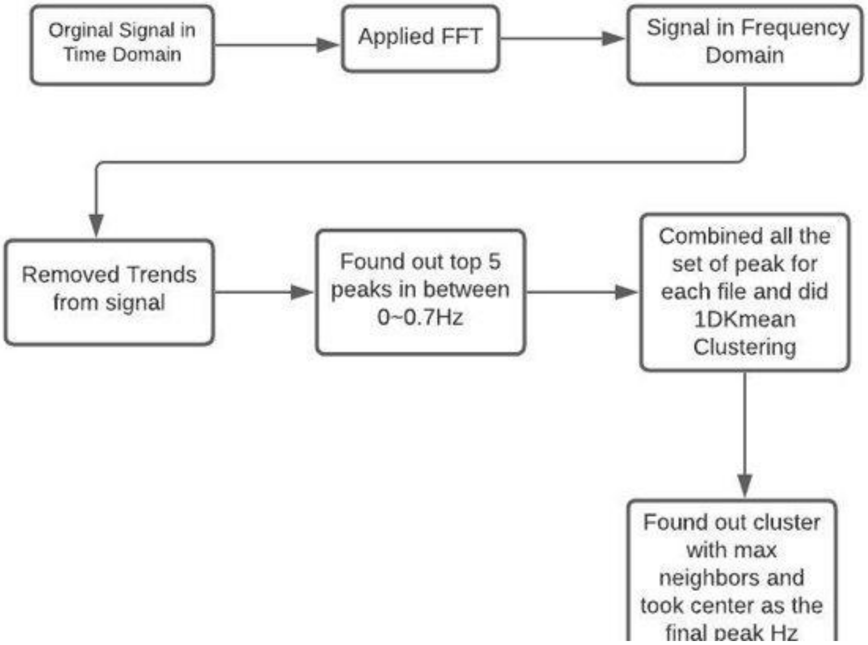

Orginal Signal in Time Domain
Applied FFT
Signal in Frequency Domain
Removed Trends from signal
Found out top 5 peaks in between 0~0.7Hz
Combined all the set of peak for each file and did 1DKmean Clustering
Found out cluster with max neighbors and took center as the final peak Hz

Figure-6a: Peak frequency determination clustering approach in brand steps

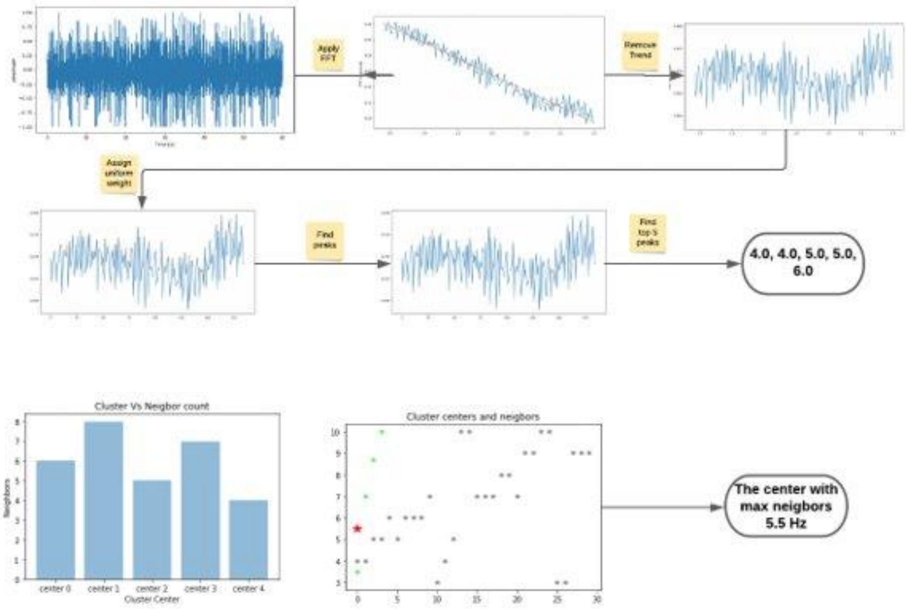


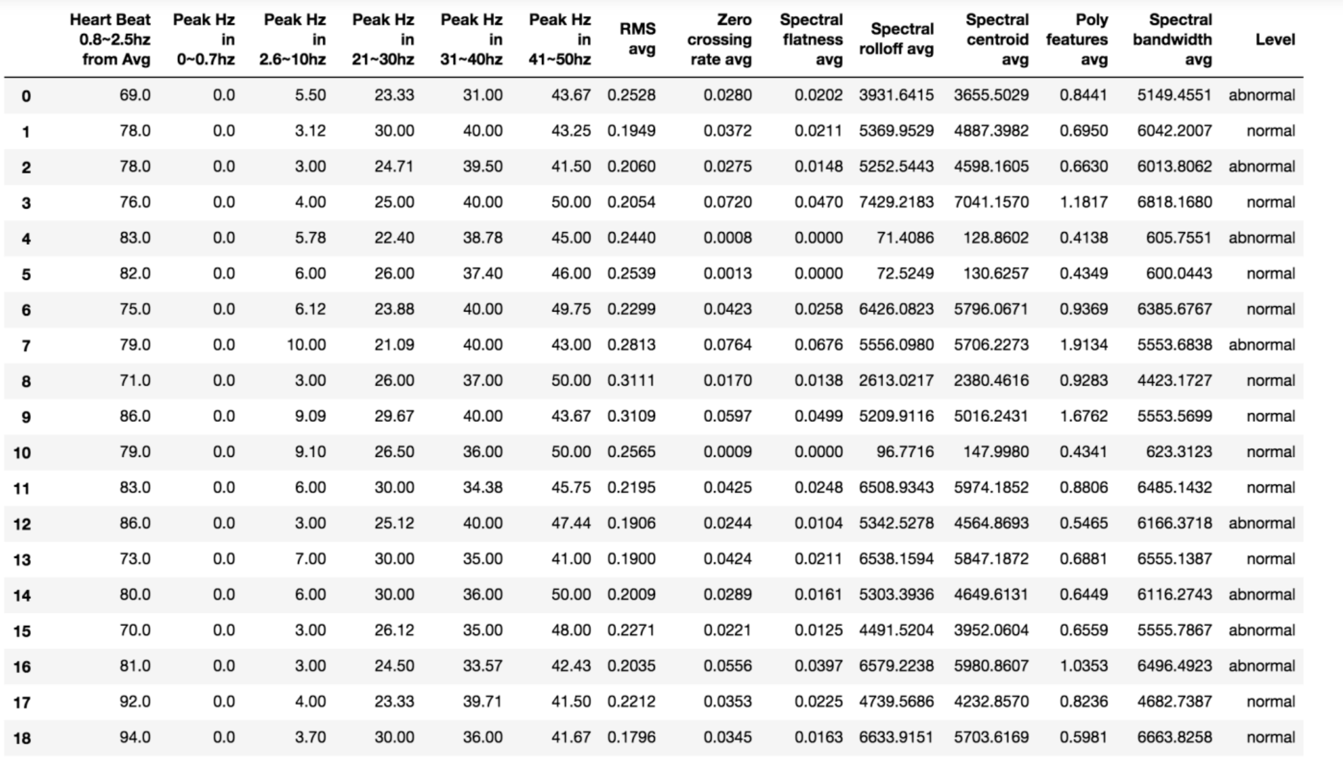
Figure-6b: Peak frequency determination clustering approach in brand

- **Feature extraction in time domain:** We extracted extract some features from original signal in time domain in a sense that those features might play important role in machine learning algorithm. Here are those features
  - **Zero crossing rate average**: Computed the zero-crossing rate of each original signal in time domain and later averaged the values over all the signal files of an individual.
  - **Spectral rolloff average:** Computed roll-off frequency of each original signal in time domain and later averaged the values over all the files of an individual.
  - **Spectral centroid average**: Computed spectral centroid of each original signal in time domain and later averaged the values over all the files of an individual.
  - **Spectral bandwidth average**: Computed spectral bandwidth of each original signal in time domain and later averaged the values over all the files of an individual.
  - **Poly features average**: Get coefficients of fitting a first-order polynomial of each original signal file and averaged the values over all the files of an individual.

- **RMS average**: Computed root means square of each original signal in time domain and later averaged the values over all the files of an individual.
- **Spectral flatness average**: Computed spectral flatness of each original signal in time domain and later averaged the values over all the files of an individual.

Extract Level from participant data

At this point we have extracted features from the signal files for each individual. We need Level to feed supervised machine learning classifier. We knew that original data file can be indexed by subId. And Participant data can also be indexed by subId. In participant data we have health issues. We used this health issues as an indication of health condition (Normal/ Not Normal). Meaning that If there is a health issue, we conclude that individual as 'Not Normal' otherwise 'Normal' By merging participant data and our extracted data using subId and derived 'Level' of each individual. By doing so, we have now 19 observations that have level as some of our data's subId is missing in participant data. Here are all extracted features with level.



	Heart Beat 0.8~2.5hz from Avg	Peak Hz in 0~0.7hz	Peak Hz in 2.6~10hz	Peak Hz in 21~30hz	Peak Hz in 31~40hz	Peak Hz in 41~50hz	RMS avg	Zero crossing rate avg	Spectral flatness avg	Spectral rolloff avg	Spectral centroid avg	Poly features avg	Spectral bandwidth avg	Level
0	69.0	0.0	5.50	23.33	31.00	43.67	0.2528	0.0280	0.0202	3931.6415	3655.5029	0.8441	5149.4551	abnormal
1	78.0	0.0	3.12	30.00	40.00	43.25	0.1949	0.0372	0.0211	5369.9529	4887.3982	0.6950	6042.2007	normal
2	78.0	0.0	3.00	24.71	39.50	41.50	0.2060	0.0275	0.0148	5252.5443	4598.1605	0.6630	6013.8062	abnormal
3	76.0	0.0	4.00	25.00	40.00	50.00	0.2054	0.0720	0.0470	7429.2183	7041.1570	1.1817	6818.1680	normal
4	83.0	0.0	5.78	22.40	38.78	45.00	0.2440	0.0008	0.0000	71.4086	128.8602	0.4138	605.7551	abnormal
5	82.0	0.0	6.00	26.00	37.40	46.00	0.2539	0.0013	0.0000	72.5249	130.6257	0.4349	600.0443	normal
6	75.0	0.0	6.12	23.88	40.00	49.75	0.2299	0.0423	0.0258	6426.0823	5796.0671	0.9369	6385.6767	normal
7	79.0	0.0	10.00	21.09	40.00	43.00	0.2813	0.0764	0.0676	5556.0980	5706.2273	1.9134	5553.6838	abnormal
8	71.0	0.0	3.00	26.00	37.00	50.00	0.3111	0.0170	0.0138	2613.0217	2380.4616	0.9283	4423.1727	normal
9	86.0	0.0	9.09	29.67	40.00	43.67	0.3109	0.0597	0.0499	5209.9116	5016.2431	1.6762	5553.5699	normal
10	79.0	0.0	9.10	26.50	36.00	50.00	0.2565	0.0009	0.0000	96.7716	147.9980	0.4341	623.3123	normal
11	83.0	0.0	6.00	30.00	34.38	45.75	0.2195	0.0425	0.0248	6508.9343	5974.1852	0.8806	6485.1432	normal
12	86.0	0.0	3.00	25.12	40.00	47.44	0.1906	0.0244	0.0104	5342.5278	4564.8693	0.5465	6166.3718	abnormal
13	73.0	0.0	7.00	30.00	35.00	41.00	0.1900	0.0424	0.0211	6538.1594	5847.1872	0.6881	6555.1387	normal
14	80.0	0.0	6.00	30.00	36.00	50.00	0.2009	0.0289	0.0161	5303.3936	4649.6131	0.6449	6116.2743	abnormal
15	70.0	0.0	3.00	26.12	35.00	48.00	0.2271	0.0221	0.0125	4491.5204	3952.0604	0.6559	5555.7867	abnormal
16	81.0	0.0	3.00	24.50	33.57	42.43	0.2035	0.0556	0.0397	6579.2238	5980.8607	1.0353	6496.4923	abnormal
17	92.0	0.0	4.00	23.33	39.71	41.50	0.2212	0.0353	0.0225	4739.5686	4232.8570	0.8236	4682.7387	normal
18	94.0	0.0	3.70	30.00	36.00	41.67	0.1796	0.0345	0.0163	6633.9151	5703.6169	0.5981	6663.8258	normal

Table-1: Extracted Feature with level

Feature Selection

Feature selection is important step in machine learning. By Feature selection we find important features those are most important in explaining the target variable. In our dataset we used RandomForestClassifier to see feature's importance. Note that we removed "Peak Hz in 0~0.7Hz" as all the values are zero. Here is the result we got by feeding our dataset.

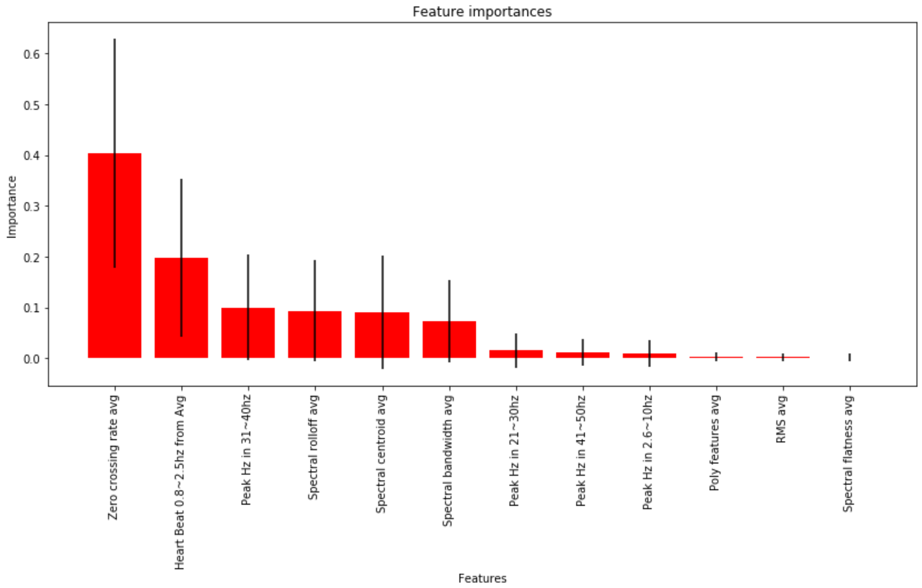


Figure-7: Feature importance

Here we can see the Zero-crossing rate average has the highest contribution, then Heartbeat and peak frequency between 31~40Hz and then spectral Rolloff frequency and so on. Some like Poly features average, RMS average and Spectral flatness average has the lowest contribution.

In order to found out top important sets of features we feed RandomForestClassifier to five different sets of top important features [6, 8, 9, 10]. and plot the accuracy, precision and recall.

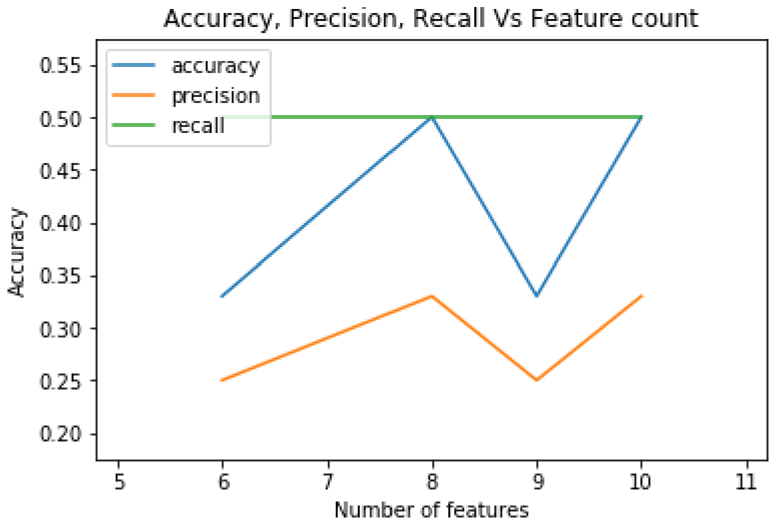


Figure-8: Accuracy, Precession and Recall for sets [6, 8, 9, 10] of features

We can see that recall are same for all, but accuracy and procession are higher in 8 sets of features. Here are the those

Feature ranking:

1. feature Zero crossing rate avg (0.403545)

2. feature Heartbeat 0.8~2.5hz from Avg (0.197096)

3. feature Peak Hz in 31~40hz (0.100103)

4. feature Spectral rolloff avg (0.092726)

5. feature Spectral centroid avg (0.091023)

6. feature Spectral bandwidth avg (0.073165)

7. feature Peak Hz in 21~30hz (0.015644)

8. feature Peak Hz in 41~50hz (0.010959)

Data Preparation for model

We modeled some supervised classification algorithms such as Random Forest, KNN, Decision Tree Classifier, Adaboost classifier to solve this problem. To be able to test the performance of our algorithms, we standardized our preprocessed clean data and then split into train test split by the ration of 0.30 and feed into model.

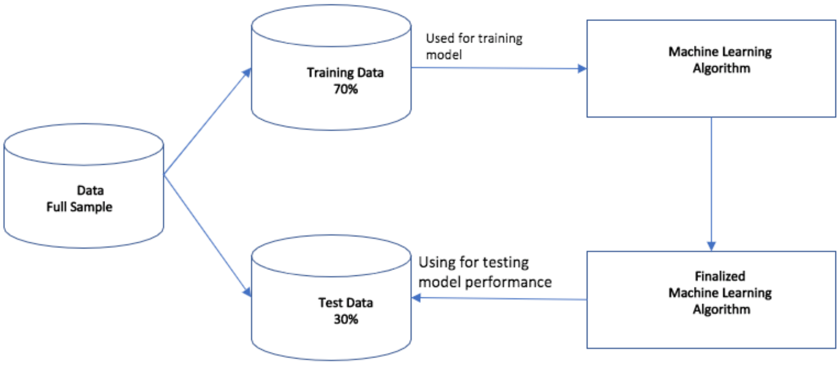


Figure-9: Train and test split for model

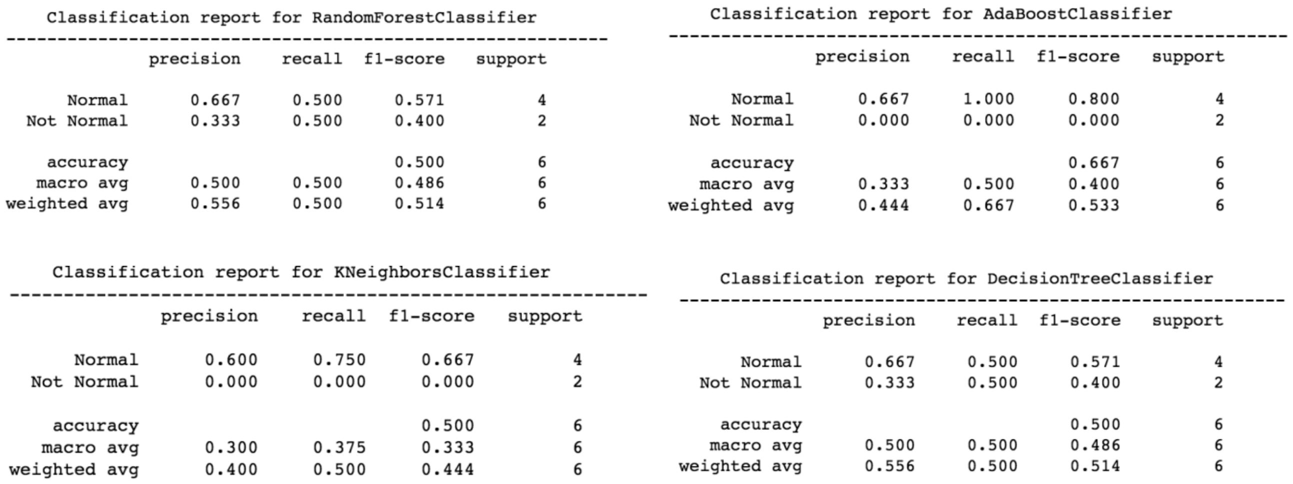
Model with Original Observations

We have total 19 observations that have level. We developed four models. those are AdaBoostClassifier, DecisionTreeClassifier, KNeighborsClassifier and RandomForestClassifier. We have spliced our data into 30% test and 7.0% train, and scaled data using StandardScaler then fit into the models. Here is the result we got.

Classifier	**Accuracy**
AdaBoostClassifier	67%
DecisionTreeClassifier	50%
KNeighborsClassifier	50%
RandomForestClassifier	50%

Table-2: Classifier vs Accuracy for original set of observation

Here is the classification report for the model



```
      Classification report for RandomForestClassifier
---------------------------------------------------------
              precision    recall  f1-score   support

      Normal      0.667     0.500     0.571         4
  Not Normal      0.333     0.500     0.400         2

    accuracy                          0.500         6
   macro avg      0.500     0.500     0.486         6
weighted avg      0.556     0.500     0.514         6
```

```
      Classification report for AdaBoostClassifier
---------------------------------------------------------
              precision    recall  f1-score   support

      Normal      0.667     1.000     0.800         4
  Not Normal      0.000     0.000     0.000         2

    accuracy                          0.667         6
   macro avg      0.333     0.500     0.400         6
weighted avg      0.444     0.667     0.533         6
```

```
      Classification report for KNeighborsClassifier
---------------------------------------------------------
              precision    recall  f1-score   support

      Normal      0.600     0.750     0.667         4
  Not Normal      0.000     0.000     0.000         2

    accuracy                          0.500         6
   macro avg      0.300     0.375     0.333         6
weighted avg      0.400     0.500     0.444         6
```

```
      Classification report for DecisionTreeClassifier
---------------------------------------------------------
              precision    recall  f1-score   support

      Normal      0.667     0.500     0.571         4
  Not Normal      0.333     0.500     0.400         2

    accuracy                          0.500         6
   macro avg      0.500     0.500     0.486         6
weighted avg      0.556     0.500     0.514         6
```

Figure-10: Model’s Classification report for original data

Then we did parameter tune by GridSeachCv on each model, but we did not much see improvement.

Model	**Accuracy**	**Best Parameters**
AdaBoostClassifier	67%	{'n_estimators': 100}
DecisionTreeClassifier	67%	{'criterion': 'gini', 'max_depth': 4, 'max_features': 4, 'splitter': 'random'}
KNeighborsClassifier	50%	{'n_neighbors': 2, 'weights': 'uniform'}
RandomForestClassifier	50%	{'criterion': 'gini', 'max_depth': 2, 'n_estimators': 100}

Table-3: Classifier vs Accuracy for original set of observation after tuning

Model with Original Observations and Important Features

We again feed the model with models with the importance features of original observations. We see the result below.

Model	**Accuracy**
AdaBoostClassifier	67%

DecisionTreeClassifier	67%
KNeighborsClassifier	50%
RandomForestClassifier	50%

Table-4: Classifier vs Accuracy for original set of observation with importance features

So, there is no significant difference impact on accuracy on each model.

Synthetic data Generation

In our data set, we only have 19 observations. So, it is very small data to apply machine learning model to it. In the fields such as medicines, sociology and psychology etc. small samples are not rare occurrence but normal. Most experimental involving primary research with real people will have small data due to cost of conduction in person. In our case we need to conduct an experimental laser scan of group of people and this collection process is too difficult and time-consuming. Small dataset models require low complexity model and avoid overfitting. So, we need to generate synthetic data to feed our model so that it does not suffers from high bias and overfit.

Data augmentation is a technique that can used to increase observations. We used GAN [1] to generate synthetic observations. GAN is generative adversarial network that can learn from training set and generate a new data with same statistic as the training set. GAN is most used in image and text data augmentation but in our case, we have tabular data. So, we used TCGAN [3] to generate 200 synthetic data to fit our models.

Model with synthetic and original data

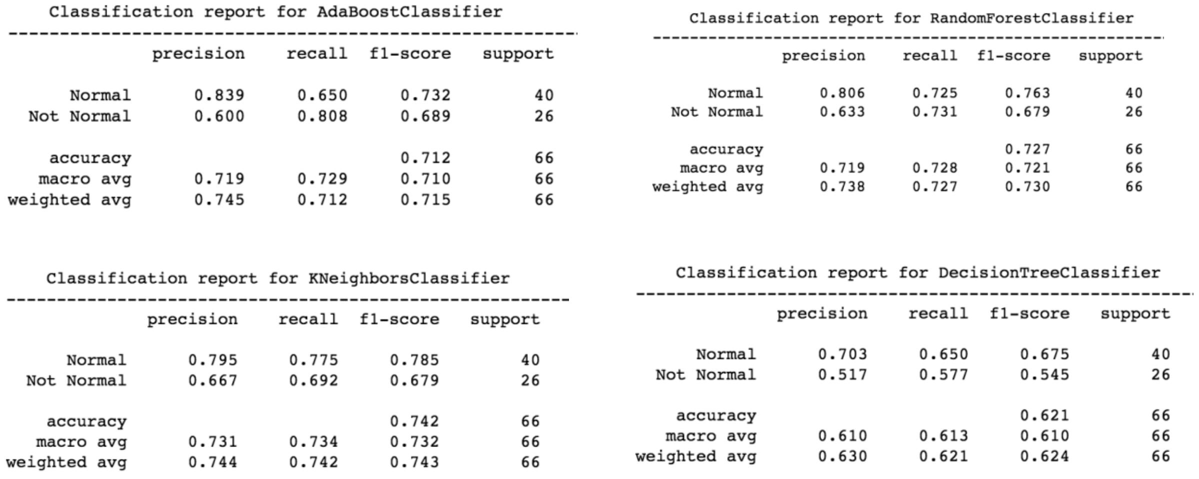
As we said before we used TCGAN [3] to generate 200 observation and we have another 19 real observations, So total 219 observations. We feed our four classifiers with this new set of data and we got accuracy little bit better than we had before. Here is the summary of model and their accuracy in table.

Model	**Accuracy**
AdaBoostClassifier	71%

DecisionTreeClassifier	62%
KNeighborsClassifier	74%
RandomForestClassifier	73%

Table-5: Classifier vs Accuracy for synthetic plus original data with importance features

We can see an improvement in KNN (24%), RandomForestClassifier(23%), decision tree is down by (5%) and AdaBoostClassifier is improved by 4%. Here is the classification report of each model on new dataset.



```
       Classification report for AdaBoostClassifier
-------------------------------------------------------
              precision    recall  f1-score   support

      Normal      0.839     0.650     0.732        40
  Not Normal      0.600     0.808     0.689        26

    accuracy                          0.712        66
   macro avg      0.719     0.729     0.710        66
weighted avg      0.745     0.712     0.715        66
```

```
       Classification report for RandomForestClassifier
-------------------------------------------------------
              precision    recall  f1-score   support

      Normal      0.806     0.725     0.763        40
  Not Normal      0.633     0.731     0.679        26

    accuracy                          0.727        66
   macro avg      0.719     0.728     0.721        66
weighted avg      0.738     0.727     0.730        66
```

```
       Classification report for KNeighborsClassifier
-------------------------------------------------------
              precision    recall  f1-score   support

      Normal      0.795     0.775     0.785        40
  Not Normal      0.667     0.692     0.679        26

    accuracy                          0.742        66
   macro avg      0.731     0.734     0.732        66
weighted avg      0.744     0.742     0.743        66
```

```
       Classification report for DecisionTreeClassifier
-------------------------------------------------------
              precision    recall  f1-score   support

      Normal      0.703     0.650     0.675        40
  Not Normal      0.517     0.577     0.545        26

    accuracy                          0.621        66
   macro avg      0.610     0.613     0.610        66
weighted avg      0.630     0.621     0.624        66
```

Figure-11: Model's Classification report for synthetic plus original data

Result Comparison

We evaluated performance of our models by computing metrics like accuracy, recall, precision, and f1 score. Here is summary of classification report in table.

	Precision		**Recall**		**F1 Score**	
Model	**Normal**	**Not Normal**	**Normal**	**Not Normal**	**Normal**	**Not Normal**
AdaBoostClassifier	83.9	60	65	80.8	73.2	68.9
DecisionTreeClassifier	70.3	51.7	65	57.7	67.5	54.5
KNeighborsClassifier	79.5	66.7	77.5	69.2	78.5	67.9

RandomForestClassifier	80.6	63.3	72.5	73.1	76.3	67.9

Table-6: summary of classification report

Here is the comparative visual representation of precision, recall and F1 Score.

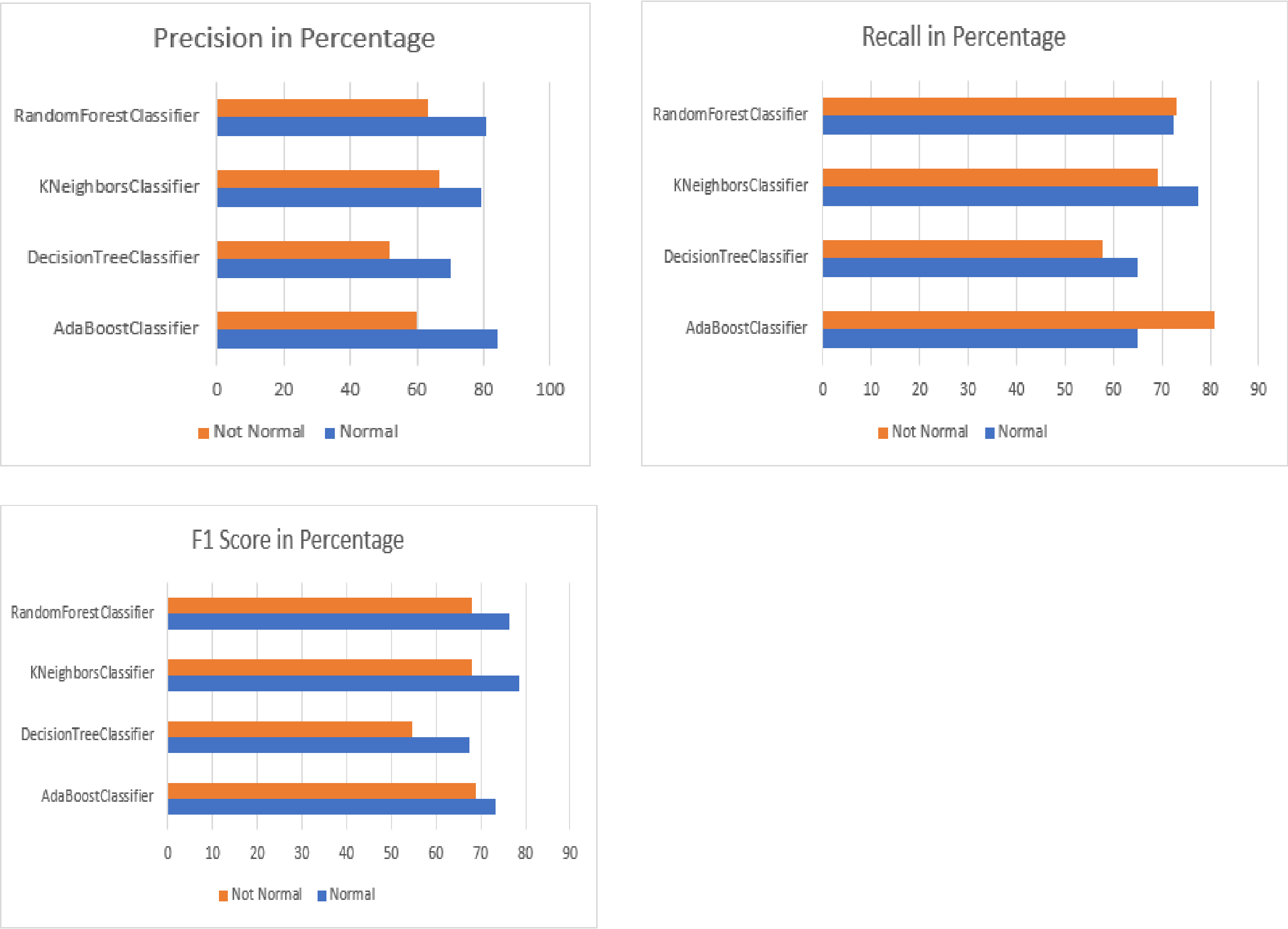


Figure-12: Model Precession, recall and F1 Score in percentage

After training all the models, Precision for AdaBoost regression is high in both categories (Normal 83.9% and Not Normal 60%) compared to other models. Next high precession model is Random forest which has precessions 80.6% on Normal and 63.3% on Not Normal category. Then KNN and Decission Tree.

If we look at recall, KNN has highest recall 77.5% on normal category and 69.2% on not normal category. Then Random forest (72.5% normal and 73.1% not normal). Then Adaboost and Decision tree.

F1-Score, KNN has highest f1-score, 78.5% on normal and 67.9% on not normal. Then Random forest, 76.3 % on normal and 67.9% on not normal category. Then Adaboost and Decision tree.

From the above discussion about precession, recall and f1-score, we can conclude that Adaboost and KNN are good compared other models. Now, let's look at accuracy of the models.

A comparative accuracy visual representation of models

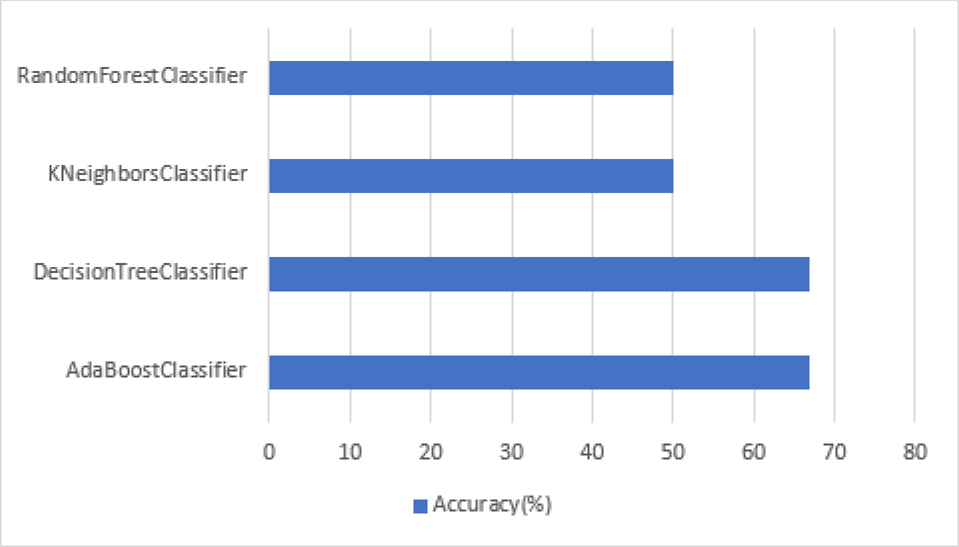


Figure-13: Models accuracy in percentage for original data

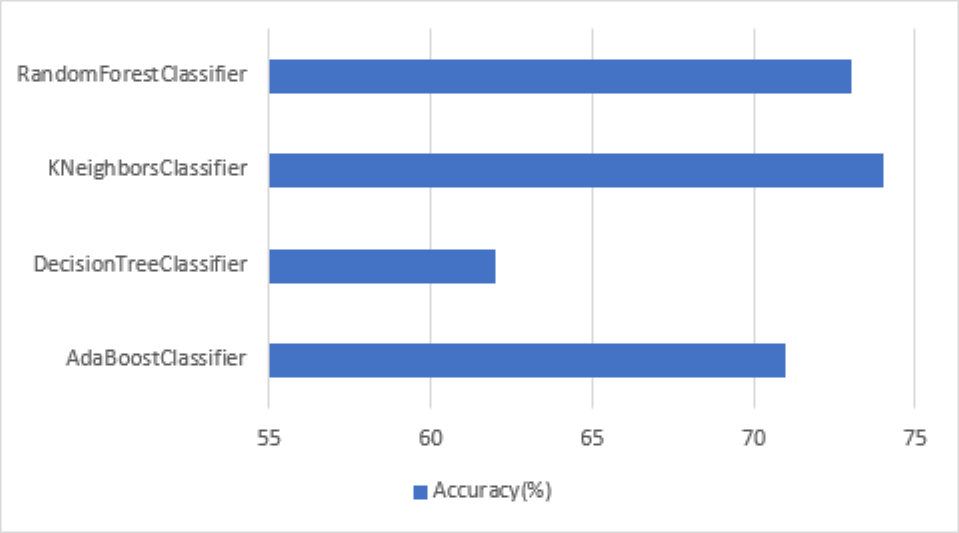


Figure-14: Models accuracy in percentage for synthetic plus original data

In the first set of accuracy of original data and best parameter tuning, we can see Decision tree and Adabooast have accuracy 67%. KNN and Random forest have 50%

In the second set accuracy of original plus synthetic data, we can see that KNN has highest accuracy which is 74% and then Random forest which has 73% accuracy. Then Adaboost which as 71% accuracy. Then Decision tree has the lowest accuracy.

After analyzing this accuracy for our models. We preferred to use Adaboost because it has a 67% accuracy on original data and 71% accuracy on original plus synthetic data. Although we KNN and Random forest has high accuracy in original plus synthetic data but they have low accuracy in original data.

Conclusion and Future Work

We have small set of data. But in machine learning, small data are not rare occurrence but normal. Most experimental involving primary research with real people have small data due to sheer cost of conduction in person. In our case this collection process is costly, too difficult and time-consuming. That's why we used to TCGAN to generate synthetic data.

We tried four different classifiers like AdaBoost, Decision Tree, KNN and Random Forest. We saw that Adaboost is the best choice.

We also learn features extracted from original signal in time domain has significant impact on machine learning.

In future we planned to extend this work to identify sex and age group of individual

Github link: https://github.com/msarker000/dse-capstone

Bibiliiography

1. https://www.kdnuggets.com/2019/06/5-ways-lack-data-machine-learning.html

2. https://en.wikipedia.org/wiki/Generative_adversarial_network

3. https://github.com/Diyago/GAN-for-tabular-data/tree/master/ctgan

4. https://arxiv.org/abs/1907.00503

5. https://www.polytec.com/us/vibrometry/technology/

6. https://www.dannyadam.com/blog/2019/07/kmeans1d-globally-optimal-efficient-1d-k-means/

7. https://pypi.org/project/kmeans1d/

8. https://www.sciencedirect.com/science/article/pii/S2352914817300242

9. https://www.kdnuggets.com/2019/06/5-ways-lack-data-machine-learning.html

10 . https://github.com/msarker000/ml-group-project

11. https://librosa.org/doc/latest/feature.html#spectral-features