# Parkinson Disease Detection

EE 660 Course Project

**Project Type:** (1) Design a system based on real-world data

**Number of student authors:** 1

Hardik Prajapati | hprajapa@usc.edu

12/06/2021

# 1. Abstract

Patients with Parkinson disease suffer nervous system issues, making it harder to draw smooth spiral and wave curves. While this disease cannot be cured but an early detection would help improve the physical condition of the patient with appropriate medication. This project is centered around three learning algorithm models, Decision-Tree, Random Forest and Adaboost to detect Disease. Since, the problem statement revolves around medical testing/screening, reasonable accuracy with high sensitivity score is key evaluation metric. Random Forest gave an accuracy of 81% with sensitivity being 71% on Spiral test. On the other hand, Random Forest performed poorly on Wave test with accuracy 61% and sensitivity 59%. We later show a Transfer Learning technique which will increase accuracy to 63% and sensitivity to 81% on Wave test.

# 2. Introduction

## 2.1. Problem Type, Statement and Goals

(1) We have a 2-class classification problem at hand. The 2 classes being Healthy (0) and Parkinson (1).

(2) We need to build a model which will predict whether the candidate giving the test has Parkinson disease or not. While maintaining the accuracy, it's also important to achieve a high sensitivity.

(3) The goal is to find the best model for each Spiral and Wave screening test.

(4) Our dataset comprises of images and hence we have a high dimension feature space with limited number of training and testing samples. With a smaller number of samples, it's difficult to build a Convolutional Neural Network from scratch. Data augmentation won't be any helpful to CNN since we might end up creating mismatched samples. Also, our approach is computationally cheaper than CNN by a large margin with decent performance.

Extension topic: Transfer Learning Approach

1) We again have a 2-class classification problem with source and target data. We make Spiral images as Source domain and Wave Images as Target Domain
2) Since Supervised learning technique did not perform well on Wave test, we use spiral images in addition to wave training images to build a model with improved results on wave screening test.
3) As the feature space is of high dimension, it's difficult to estimate the density of both source and target domains. In transfer learning approach, working with CNN is more effective by using pre-trained

models and fine tuning it. Still, it's computationally more expensive than our approach.

## 2.2. Literature Review (Optional)

[1] Zham et al. proposed a study that used two criteria such as speed, and pen-pressure while performing the sketches to distinguish PD subjects at different stages. [2] Kotsavasiloglou et al., presented an investigation based on the trajectory of the tip of the pen on the surface of the pad while drawing simple horizontal lines by the healthy subjects and PD subjects.

## 2.3. Our Prior and Related Work (Mandatory)

"Our Prior and Related Work - None".

## 2.4. Overview of Our Approach

### 2.4.1 Main topic

The baseline systems are

- The trivial baseline: Classifier that outputs 'Parkinson' always
- The non-trivial baseline: Decision Tree

In this section, we discuss Decision Tree, Random Forest and Adaboost classifiers which were used in this study. Decision Tree classifier follows a tree like structure where each node of the tree represents best split of the previous region. If features are known, then this classifier is human interpretable. Random forest and Adaboost are ensemble algorithms. Random Forest is build comprising of many Decision Tree and hence the collective output is better than any individual tree. Adaboost is a type of forward stagewise Additive modelling, where it runs a base model for multiple times and ultimately outputs the weighted sum of all the learners. Evaluation metric comprises of Accuracy, Sensitivity and Specificity. We compare the performance of Random Forest and Adaboost with the base classifier and expect them to perform better than both base classifiers. We plot bar charts for all the above metrics for both Spiral and Wave Image datasets.

### 2.4.2 Extension (Transfer Learning)

The baseline systems are

- Trivial: Model trained on Spiral (source) images and tested on Wave(target) testing images.

- Non-Trivial: Model trained on union of Spiral and Wave training data and tested on Wave testing images

In this section we discuss the Transfer Learning approach using Importance weighting. We use Decision tree, Random Forest, Adaboost with sample weights to train on the union of source (train + test) and target (train) data and evaluate Accuracy, Sensitivity and Specificity metric. Importance weighting technique applies a penalty for drawing data from source domain to predict on target domain data. We compute class conditional probability density for domain using 2 separate techniques i) Gaussian Mixture ii) Kernel density estimation (kernel=Gaussian). The weights are calculated as follows:

$$x_s \neq x_{T;}\, P(y|x_s) \neq P(y|x_T);\; p(y_s) = P(y_T)$$

Weights = $P_T(x|y) / P_S(x|y)$ for all $x \in (x_s$(train,test) + $x_T$ (train))

# 3. Implementation

## 3.1. Data Set

We have 2 datasets: Spiral Image dataset and Wave Image dataset [3]. Each dataset contains 72 training samples and 30 testing samples. We have a balanced dataset in training and testing directories for both spiral and image datasets. The extracted features are real-valued data, and the cardinality of feature space is 12296. We use OpenCV library to read images and convert them to grayscale before extracting the features.

## 3.2. Dataset Methodology

We did not have the provision of Validation dataset. We worked with only training and testing datasets. We trained the models on Training dataset and estimated out-of-sample accuracy, sensitivity and specificity using test datasets. We used test dataset to compare different models with regards to the evaluation metrics

## 3.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

Preprocessing involved converting images to grayscale and resizing them to 200x200 using OpenCV library functions resize & cvtColor. We extracted Histogram of gradient features using skimage.feature function which basically quantifies each image. Dimensionality reduction happened at 2 stages: a) resizing the image b) reducing the number of channels from RGB to grayscale. We transform the target labels to (0,1) class by applying label encoder function of sklearn.

## 3.4. Training Process

Trivial Model: All Parkinson

- This model outputs Parkinson class every time.
- Simple Methods for baseline model, since it's important that there are very few False Negative.
- There are no parameters for this model
- This model has no complexity.

Non-Trivial Model: Decision Tree

- Decision Tree classifier builds a tree like structure, with every node indicating a decision made based on the best split of the previous region. We use sklearn's DecisionTreeClassifier() function to fit on the training and labeled data. At each node, it splits the previous region into 2 sub-regions by optimizing the loss function. This includes choosing a feature, finding the threshold, labeling the samples and computing classification error.
- This classifier is a human-interpretable model if we have feature information. Also, we wanted to implement ensemble models, hence Decision Tree provides a good benchmark for evaluating the ensemble models.
- The model was fitted with the default parameters as this was a baseline model. The model is run for 5 trials, and we take mean for each metric values.

Random Forest:

- Random Forest Classifier is an ensemble type modelling which builds a concentration of decision trees. If we consider N trees as N random Variables, then the Random Variable representing the average sum of these N trees will have a reduced variance. We use sklearn's RandomForestClassifier() function to fit on training data.
- Random Forest is built upon a bunch of trees and hence performs better than any single individual tree. This classifier learns from all the trees and outputs a weighted sum of these trees.
- We assigned the parameter, n_estimators = 100 which declares the total number of decision trees to be incorporated. Since we did not have provision of any validation dataset, we chose the value from our previous knowledge and experience working with this classifier.

- We run the model for 5 trials and compute the mean of each metric values.

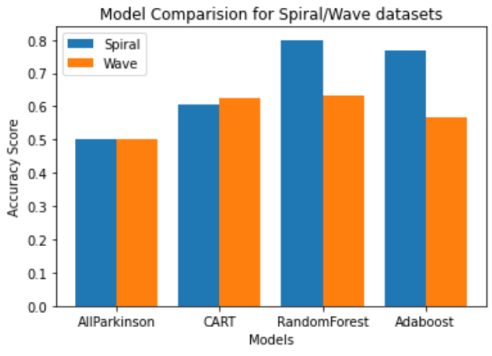
Adaboost:

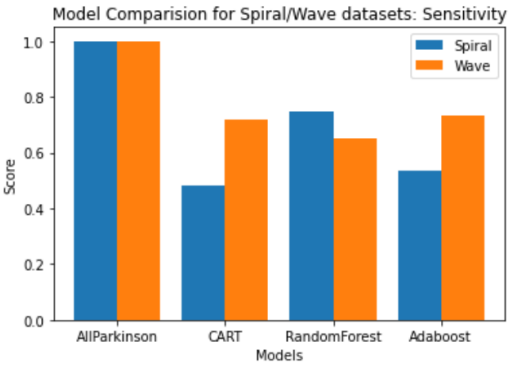
- Adaboost is another ensemble forward stagewise additive modelling technique. This classifier runs B base models and computes a weighted sum of it. While learning, the classifier gives importance (more weight) to misclassified samples in b-1 classifier for computing the model of $b^{th}$ learner. We use sklearn's AdaboostClassifier() function to fit on the training data.
- Ensemble type of learning algorithms generally perform well since there is a lot of rigorous learning running behind. The objective function for each learner weighs importance according to the error generated in the previous learner.
- We assigned 2 parameters for this model, a) n_estimators =100 b) learning_rate=0.001. n_estimators define the number of learners to be trained (B) and learning rate defines the rate of learning. Since we did not have provision of any validation dataset, we chose the value from our previous knowledge and experience working with this classifier.

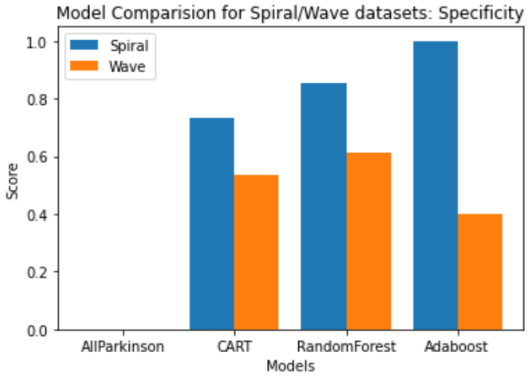
### 3.5. Model Selection and Comparison of Results

Since, we did not have the luxury of Validation dataset, there were no model selection step involved. We present the comparision of both models with the baseline models in the next section.

## 4. Final Results and Interpretation

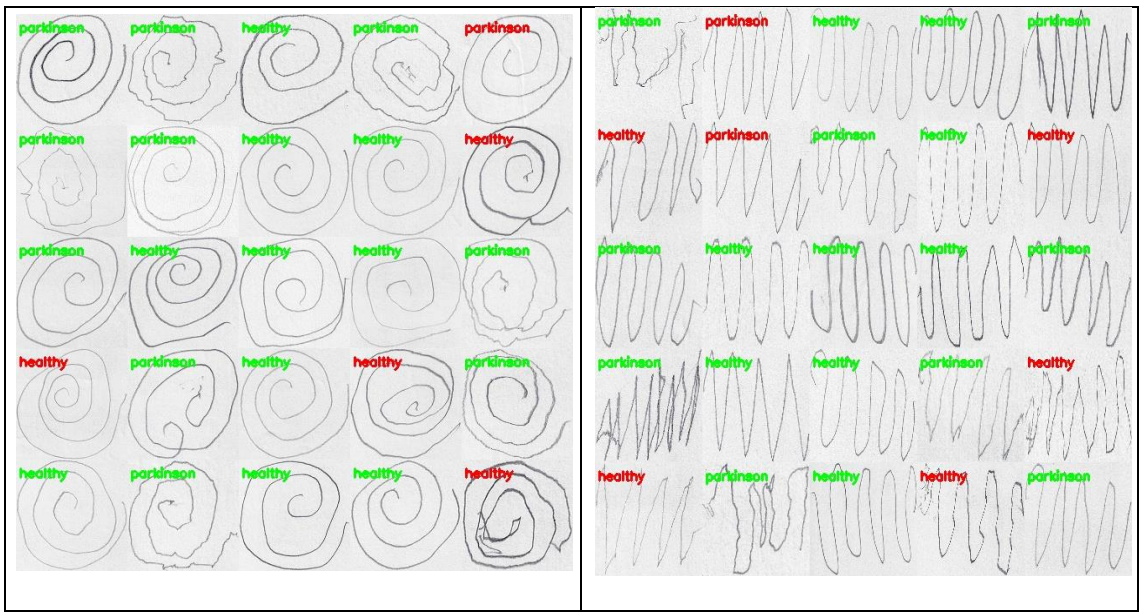


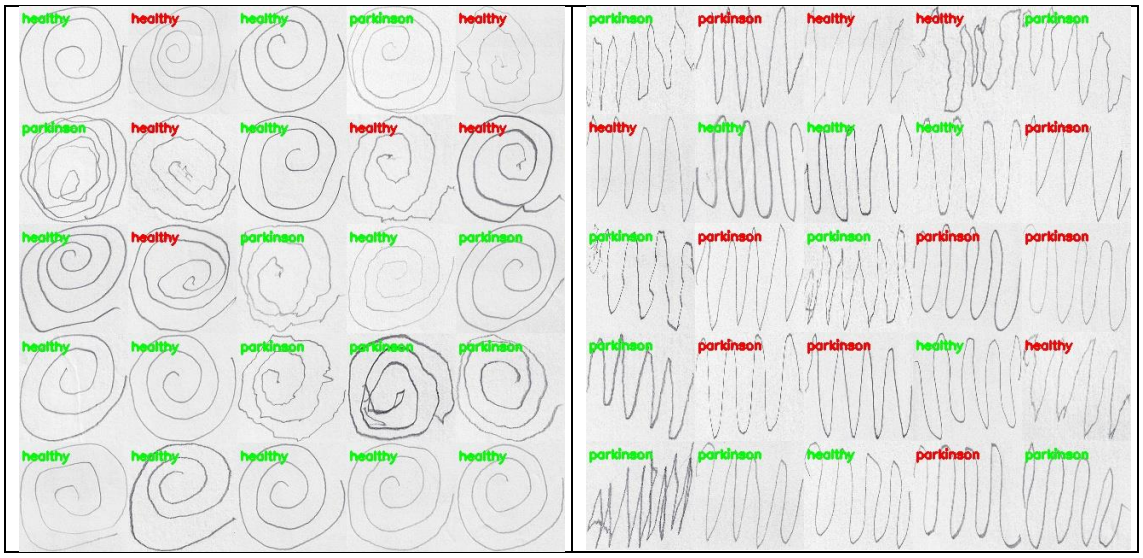



| | Spiral Dataset | | | Wave Dataset | | |
|---|---|---|---|---|---|---|
| | Accuracy | Sensitivity | Specificity | Acc | Sen | Spec |
| **All Parkinson** | 0.5 | 1.0 | 0.0 | 0.5 | 1.0 | 0.0 |
| **Decision Tree** | 0.61 | 0.48 | 0.73 | 0.62 | **0.72** | 0.53 |
| **Random Forest** | **0.81** | **0.74** | 0.85 | **0.63** | 0.65 | 0.61 |
| **Adaboost** | 0.77 | 0.53 | 1.0 | 0.56 | 0.73 | 0.4 |

**Prediction: Random Forest**



**Prediction: Adaboost**



Interpretation:

Spiral Dataset: It can be observed that Random Forest performed the best on Spiral images with accuracy of 81% and sensitivity of 74%. This means that, the model was able to predict 74% times correctly that a patient had Parkinson Disease. On the other hand, Adaboost was able to achieve 77% accuracy but very poor sensitivity (53%). In fact, Adaboost scores specificity of 100%, which means that there was no such time that it classified patient having disease but actually didn't. But our objective lays more importance on detecting Parkinson disease rather than classifying the candidate that he/she did not have. Clearly our ensemble models performed better than the baseline models.

Wave dataset: There is no clear winner over here. Rather comparatively our baseline model, Decision tree performs reasonably better in terms of both accuracy and sensitivity. We would have expected Random Forest to perform better on wave dataset too but that is not the case. This can be reasoned maybe, because the healthy candidate's drawing of waves aren't that smooth/clean and hence difficult to differentiate between the two classes. Also, for waves, the patterns are repetitive and not much variation from one cycle to other which was not the case in spirals where the diameter of curves changed. This can also be reasoned that, patients with Parkinson might face less trouble in horizontal movement of hands than vertical movements.

# 5. Implementation for the extension

## 5.1. Data Set

We have 2 datasets: Source domain (Spiral Image dataset) and target domain (Wave Image dataset). Each dataset contains 72 training samples and 30 testing samples. We have a balanced dataset in training and testing directories for both spiral and image datasets. The extracted features are real-valued data, and the cardinality of feature space is 12296. We use OpenCV library to read images and convert them to grayscale before extracting the features.

## 5.2. Dataset Methodology

We did not have the provision of Validation dataset. We worked with only training and testing datasets. We trained the models on union of Source (train,test) and target (train) dataset and estimated out-of-sample accuracy, sensitivity and specificity using test datasets. We used test dataset to compare different models with regards to the evaluation metrics

## 5.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

Same as Supervised learning case.

## 5.4. Training Process

Trivial Models: Random Forest (SL method)

- This is the supervised learning technique where we train the model in source domain and predict the target domain data.
- This is a good comparison for Supervised Learning technique and Transfer Learning technique.
- The parameters are same as for Supervised Learning technique.
- We run the model for 5 trials and compute the mean of each metric values.

Non-Trivial Model: Random Forest (TL technique)

- This is the transfer learning technique where we train the model on union of source and target domain train data and predict the target domain data.
- This is a good comparison for Transfer learning technique without using sample weights (penalty) and Transfer Learning technique with using sample weights (penalty).
- The parameters are same as for Supervised Learning technique.
- We run the model for 5 trials and compute the mean of each metric values.

Random Forest:

- Everything remains the same as in case of Supervised Learning except for 2 major changes:
    - Training data: Union of Source + Target train data
    - Model is provided with an additional parameter, sample_weights in model.fit() method.

Adaboost:

- Everything remains the same as in case of Supervised Learning except for 2 major changes:
    - Training data: Union of Source + Target train data
    - Model is provided with an additional parameter, sample_weights in model.fit() method.

We present 2 ways of computing the weights:

- Gaussian Mixture
  - For each class and each domain:
    - We reduce the dimension of data by using sklearn's PCA function.
    - We then fit Gaussian mixture model using sklearn's function and estimate density parameters of our data.
    - Using this expected density parameters we compute the PDF values of data.
- Kernel Density
  - For each class and each domain:
    - We reduce the dimension of data by using sklearn's PCA function.
    - We then fit Kernel Density model using sklearn's function with kernel=Gaussian and estimate density parameters of our data.
    - Using this expected density parameters, we compute the PDF values of data.

### 5.5. Model Selection and Comparison of Results

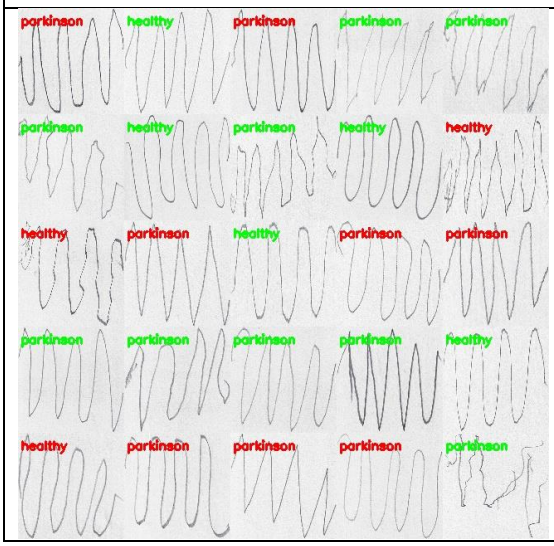
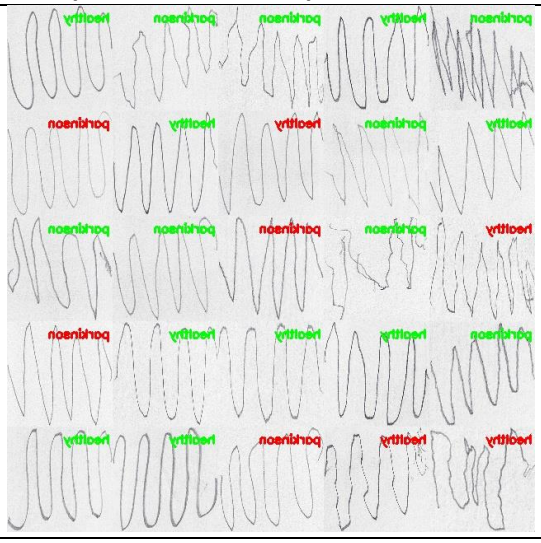
Since, we did not have the luxury of Validation dataset, there were no model selection step involved. We present the comparision of both models with the baseline models in the next section.

## 6. Final Results and Interpretation for the extension

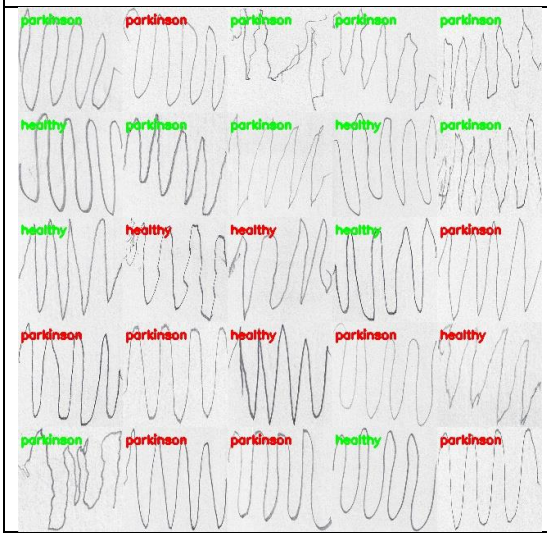
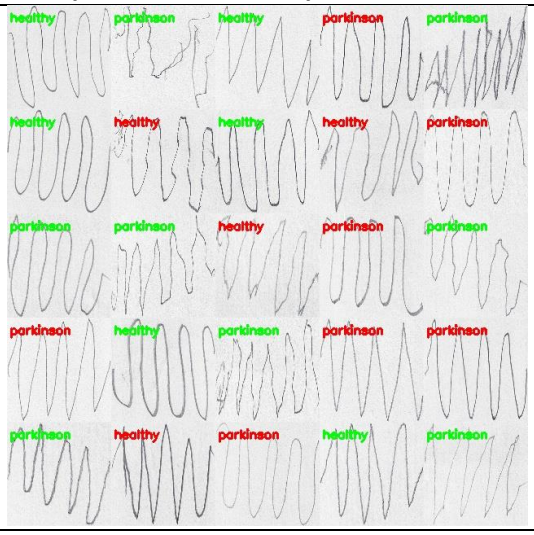
| Gaussian Mixture | **Accuracy** | **Sensitivity** | **Specificity** |
|---|---|---|---|
| **Random Forest (from section 4)** | **0.63** | **0.65** | 0.61 |
| **Trivial Base model** | 0.51 | 0.99 | 0.03 |
| **Non-Trivial** | 0.63 | 0.77 | 0.49 |
| **Random Forest** | **0.64** | **0.79** | 0.49 |
| **Adaboost** | 0.57 | 0.73 | 0.40 |

| KDE (kernel=gaussian) | **Accuracy** | **Sensitivity** | **Specificity** |
|---|---|---|---|
| **Trivial Base model** | 0.51 | 0.99 | 0.03 |
| **Non-Trivial** | 0.63 | 0.77 | 0.49 |
| **Random Forest** | 0.61 | 0.64 | 0.59 |
| **Adaboost** | 0.57 | 0.73 | 0.40 |

**Prediction: Random Forest**

| Gaussian Mixture | KDE (kernel =Gaussian) |
|---|---|
|  |  |

**Prediction: Adaboost**

| Gaussian Mixture | KDE (kernel =Gaussian) |
|---|---|
|  |  |

Interpretation:

Gaussian Mixture: Definitely, there have been good improvement of Random Forest on Wave dataset with the help of Gaussian Mixture technique. From the output montage, it's quite evident that model has started recognizing the patterns in waves and differentiating between Parkinson and healthy class. Although there hasn't been much improvement in accuracy but there has been significant improvement in sensitivity which means that, the chances of person not getting detected with

disease when he/she actually has had reduced. There has been a trade off with specificity being reduced, but it doesn't matter much unless the accuracy is also compromised. On the other hand, Adaboost hasn't shown many changes in performance. The improvement in results of random forest can also be stretched to the fact that density estimation through Gaussian Mixture was pretty good.

KDE (kernel=Gaussian): This technique of density estimation and hence the weights haven't shown good results. Although the parameters of Gaussian kernel were the same as of Gaussian Mixture, we cannot justify why this would be the case.

## 7. Contributions of each team member

"Not a team project"

## 8. Summary and conclusions

We presented Supervised Learning and Transfer Learning technique to classify candidate as Healthy/Parkinson. We presented the evaluation of each model on 2 metrics but laid more importance on Accuracy and sensitivity. We achieved Random Forest classifier with high accuracy and sensitivity of 81% and 74% respectively on Spiral Image dataset. Since Supervised Learning did not work well on Wave image dataset, we presented the technique of Transfer learning with Importance Weighting and showed improvement of Random Forest classifier on Wave datasets. We observed that Gaussian Mixture model was a good way of estimating density. We would have liked to have more data in order to have the luxury of validation dataset and hence perform model selection wherever necessary. We also conclude that our models are computationally cheaper and find the best fit for memory restricted embedded devices. It would be wonderful to try other transfer learning techniques to boost the performance of the models. Also, CNN with 2 parallel heads, one each for spiral and wave datasets and then combining the weightage sum of both might be interesting to implement.

## 9. References

[1] Zham, P., Arjunan, S.P., Raghav, S. and Kumar, D.K., 2017. Efficacy of guided spiral drawing in the classification of Parkinson's disease. IEEE journal of biomedical and health informatics, 22(5), pp.1648-1652.

[2] Kotsavasiloglou, C., Kostikis, N., Hristu-Varsakelis, D. and Arnaoutoglou, M., 2017. Machine learning-based classification of simple drawing movements in Parkinson's disease. Biomedical Signal Processing and Control, 31, pp.174-180.

[3] Zham P, Kumar DK, Dabnichki P, Poosapadi Arjunan S and Raghav S (2017) Distinguishing Different Stages of Parkinson's Disease Using Composite Index of Speed and Pen-Pressure of Sketching a Spiral. Front. Neurol. 8:435. doi: 10.3389/fneur.2017.00435

# 10. Appendix

Appendix I.

- You have to create a "main" file that outputs test result if we run the "main" file.
- "main" file should be named "main.py".
- "main" file should be located in the base directory of your extracted zip folder.
- If dataset is too big to be uploaded (size > 50 MB), you can provide URL link with "wget" linux command. (Please see the example below)
- The "main" file should read the data files in "data" folder and output the results of your final system in your report.
- You should not let "main" file to start training. "main" file should load trained parameters and reproduce the test results.

Example of extracted zip folder:

```
├── main.py (or main.sh – Mandatory file)
├── modules.py
├── utils.py
├── data
│   ├── data1_training_data.csv
│   ├── data1_validation_data.csv
│   ├── data1_test_data.csv
│   ├── data2_training_data.csv
│   ├── data2_validation_data.csv
│   └── data2_test_data.csv
```

if dataset is too big to be downloaded:

```
├── main.py (or main.sh – Mandatory file)
├── modules.py
├── utils.py
├── data
│   └── download.sh
```

in “download.sh” you specify the download link as in the following example:

```
wget https://archive.ics.uci.edu/ml/machine-learning-
databases/haberman/haberman.data
wget https://archive.ics.uci.edu/ml/machine-learning-
databases/haberman/haberman.names
```