Early Diagnosis of Parkinson's Disease via keystrokes

Chin Min Tee
February 11, 2018

Definition

Project Overview

Parkinson's disease sickened over 6 million individuals worldwide. The impact extended to their immediate families and friends, societies, medical communities and many more resources are not negligible.
This neurodegenerative disease is incurable but early discovery indisputably benefit the wellbeing of the patient and all involved. Diagnosis typically relies on neurologists but it is the individual patient (or close family member) who has to first discover that he or she is having the symptom of Parkinson's disease. When the well-known symptom of movement disorder becomes obvious to the patient it typically has been five to ten years since the patient has the disease. More than half of the area affected in the brain are damaged and many episodes of falls and injuries have already occurred. It is certainly more stressful to care for a patient with Parkinson's disease and physical injuries and some other form of damages like depression compare to just the disease itself.

A cost-effective test that can be done at home with simple and familiar apparatus would be one of the best solution for early detection.

Experiment has been conducted to use keyboard stroke as a mean to allow non-clinical personnel to make a guided diagnosis. This help captures the pre-motor phase of the disease, which could be years, or decades before the degeneration and the tell-tale symptoms of the failing motor control.

Inspired by the study 'High-accuracy detection of early Parkinson's Disease using multiple characteristics of finger movement while typing' recorded in physionet.org, I am eager to put machine learning to work to build a great classification model so that precious opportunity to start treating or researching the disease is not lost.

Problem Statement

The goal is to take in data sets provided by physionet.org for the study 'High-accuracy detection of early Parkinson's Disease using multiple characteristics of finger movement while typing' (Tappy keystroke data) and build a satisfactory binary classification model out of it. The model will be able to predict by using patient's keystroke data input if he or she has Parkinson's disease.

There is another project also hosted by physio.net: neuroQWERTY MIT-CSXPD. The way keystroke data was acquired is different from the Tappy project but the difference is in the soft wares and environments set up. The data collected by neuroQWERTY have the similar attribute that Tappy project collected. The preliminary exploration has shown that they can be used together with Tappy data set thereby increasing the total volume of available data.

Evaluation Metrics

Confusion matrix and F1 score will be used to evaluate models instead.

		Predicted	
		0	1
Actual	0	True Negative (**TN**)	False Positive (**FP**)
	1	False Negative (**FN**)	True Positive (**TP**)

Table 1. Confusion Matrix

Accuracy is intuitive and easy to understand. It is the correct predictions over the total predictions. However, it **<u>would not be suitable</u>** here since we have around 67 percent of data with positive Parkinson's disease. In this situation where majority belong to one of the two outcomes, the naïve model could achieve seemingly high performance by predicting everyone to have PD. There will be a lot of false positive, or Type 1 error. Even though it is preferable to err on Type I error than Type II error (False negative, result in missing treatment opportunity), there is better way to measure our model.

$$\boldsymbol{Accuracy} = \frac{\mathbf{TP + TN}}{(\boldsymbol{TP + TN + FP + FN})}$$

Precision is the correct positive predictions over all positive predictions. That is when a patient is predicted to have Parkinson's disease, how correct the prediction is. In this project, it is preferable to have high precision, which means there is low false positive, low rate of telling people falsely that they have the disease.

$$\boldsymbol{Precision} = \frac{\mathbf{TP}}{(\boldsymbol{TP + FP})}$$

Recall (also known as **Sensitivity**, or **True Positive Rate (TPR)**). When a patient is truly having Parkinson's disease, how often a model will predict such.   In this project, this is the key measurement. The higher the model could achieve a recall rate, the better the model can detect the disease if a patient does have the disease, thereby ensuring proper and prompt treatment.

$$\boldsymbol{Recall} = \textbf{Sensitivity} = \textbf{TPR} = \frac{\textbf{TP}}{(\boldsymbol{TP} + \boldsymbol{FN})}$$

Specificity explains that when a patient truly does not have Parkinson's disease, how often the model predicts such.

$$\boldsymbol{Specificity} = \frac{\textbf{TN}}{(\boldsymbol{TN} + \boldsymbol{FP})}$$

F1 score is a weighted average of precision (P) and recall (R).

$$\boldsymbol{F1} = \frac{\textbf{2} * \textbf{PR}}{(\boldsymbol{P} + \boldsymbol{R})}$$

FPR is False Positive Rate. When a patient truly has no Parkinson's disease, how often the model predicts there is Parkinson's disease.

$$\boldsymbol{FPR} = \frac{\textbf{FP}}{(\boldsymbol{TN} + \boldsymbol{FP})} = \boldsymbol{1} - \boldsymbol{Specifity}$$

ROC (AUC) is Receiving Operating Characteristics curve and AUC is area under curve. ROC curve is a plot of true positive rate (TPR) vs false positive rate (FPR). The area under curve is an indicator on how well the model is. AUC approaching 1 is considered good and near zero is bad.

Due to the understanding that the majority of population will not have Parkinson's disease, the data will be imbalanced and the metric to measure a model will need to be agnostic of the underlying bias of the data set.

One of PRC and ROC/AUC big difference is that PRC does not use TN in its calculation, whereas ROC/AUC FPR(X axis) uses TN in its calculation. It is found that when there is imbalanced data involved where TN is huge, ROC does not reflect the true metric measurement. The ROC curve stays pretty much as before. The PRC will show degradation in performance of model since FP starts to grow. At the same time, PRC is not affected by the huge TN seen, thus it is more robust in the occasion of imbalanced data.

The major metric used here will be F1 score and PRC. ROC/AUC will also be used as reference. Same set of data is used throughout the modeling and it is still useful to see the ROC/AUC of different models.

Analysis

Data Exploration

There are 2 datasets from two different sources that we will combine to form a larger dataset. The two sources are both from physionet.org, one is 'Tappy Keystroke Data' (Tappy hereafter) and the other is 'neuroQWERTY MIT-CSXPD Dataset' (NQ hereafter).

In the Tappy dataset, there are two files that need to be zipped together to form pandas DataFrame for modeling. There are ArchivedUsers.zip (User files) and ArchivedData.zip (keystrokes data files pertaining to users in User files). The User file has 10 character code as filename that represent an individual user, that 10 character code is also present in corresponding keystroke data file for that user. A user have more than one keystroke data files.

The User file contains the following:

- Birth Year: Year of birth
- Gender: Male/Female
- Parkinsons: Whether they have Parkinson's Disease [True/False]
- Tremors: Whether they have tremors [True/False]
- Diagnosis Year: If they have Parkinson's, when was it first diagnosed
- Whether there is sidedness of movement [Left/Right/None] (self reported)
- UPDRS: The UPDRS score (if known) [1 to 5]
- Impact: The Parkinsons disease severity or impact on their daily life [Mild/Medium/Severe] (self reported)
- Levadopa: Whether they are using Sinemet and the like [Yes/No]
- DA: Whether they are using a dopamine agonist [Yes/No]
- MAOB: Whether they are using an MAO-B inhibitor [Yes/No]
- Other: Whether they are taking another Parkinson's medication [Yes/No]

The Data file:

- UserKey: 10 character code for that user
- Date: YYMMDD
- Timestamp: HH:MM:SS.SSS
- Hand: L or R key pressed
- Hold time: Time between press and release for current key mmmm.m milliseconds
- Direction: Previous to current LL, LR, RL, RR (and S for a space key)
- Latency time: Time between pressing the previous key and pressing current key. Milliseconds
- Flight time: Time between release of previous key and press of current key. Milliseconds

The field Parkinsons is the label for the Tappy dataset, and the rest would eventually be features. The medicine mitigating the symptoms is to show that the patient is prescribed to have the indicated medicines, but the data collected here is done in such a way that the medicine does not influence the keyboarding activities. Considered its nature, those medicine fields will be discarded.

As for the direction, it is a feature that could complicate the modeling without further investigation into medical domain knowledge of sidedness between Parkinson's disease patient and user with preferred side. There is also no such information in NQ dataset. Thus, the decision to not use this feature at this time is made.

The data file logged all 8 fields in a row and there could be over two hundred thousand (200,000) rows per user depending on the length of the keyboarding activity. There are latencies that exceeds a certain threshold and the row will be treated as irrelevant. Currently such latencies is treated as outliers and will be ruled out in individual keystroke data file. The method use here is the Tukey's Method.

As for the NQ dataset, the data consists of the following:

- The key pressed.
- The hold duration in seconds.
- The key release time in seconds from time 0.
- The key press time in seconds from time 0.

The 'hold duration' is the same type of information with the 'Hold time'. The 'Flight time' will need to be calculated in this test file from 'key release time' and 'key press time'. These times are the timer time stamp of the keystroke when it happened. Imagine a timer device in a track race is zeroed out and started counting upwards when activity begun. To get the 'Direction' from this test set, it needs some calculation as well from the 'key pressed' from two adjacent rows.

The Tappy dataset has 227 users, but only 217 users has actual activities. And Among those, 48 are healthy and 169 have Parkinson's disease.

For the NQ dataset, there are two sets of experiments, PD_MIT-CS1PD and PD_MIT-CS2PD. When combined they have 85 users, 43 are healthy and 42 have Parkinson's disease.

The combined dataset has 302 users. It was later than found out that 2 of the Tappy users only has one and two keystroke records and they are discarded. This results in 300 users with 203 contracted Parkinson's disease and 97 healthy.

The positive disease prevalence rate is 203/300, which is 67.67%.

Benchmark

A crude, simple model like default parameterized support vector machine classifier or decision tree model could blatantly predict that everyone has Parkinson's disease and the precision would be

203/300 which is 67.7%, and recall will top at 100%. F1 score is logged at 80%. But the ROC AUC will suffer at 50%, which is useless as the model lose the ability to tell the difference.

Confusion Matrix	TN=5	FP=16
	FN=3	TP=36
F1	0.79	
Precision	0.6923	
Recall	0.9231	
ROC/AUC	0.6447	

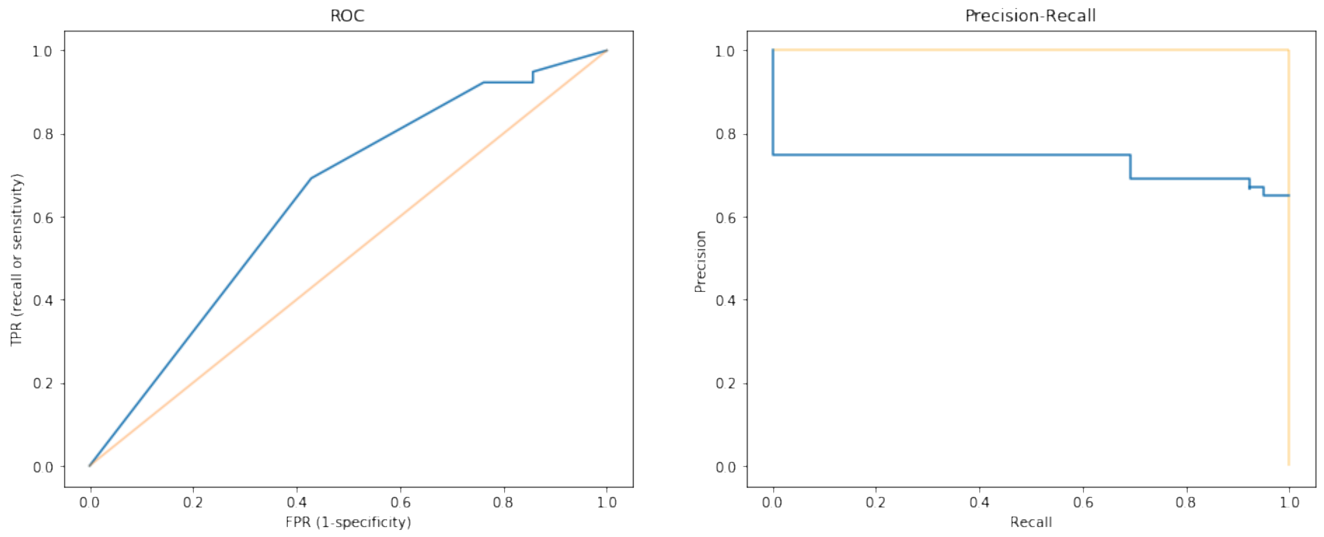


Figure 1 Result for benchmark Decision Tree classifier

Exploratory Visualization

Two individual combined files were randomly picked for exploratory activities.

Hold times for both PD group and no PD group are plotted below, and the hue separated out the direction from one key to another. Notice that they is no obvious trend when going from one zone of keyboard to another. However, there is a trend that no PD group has a somewhat lower hold time of approximately ~100ms, compare to PD group with ~200ms.

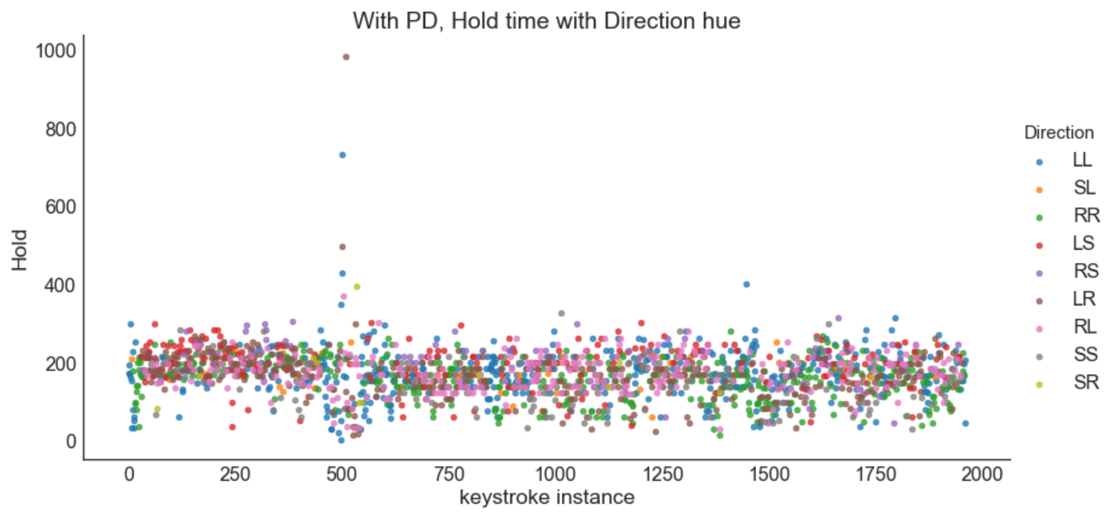


Figure 2. Exploratory Hold time plot for a user with PD

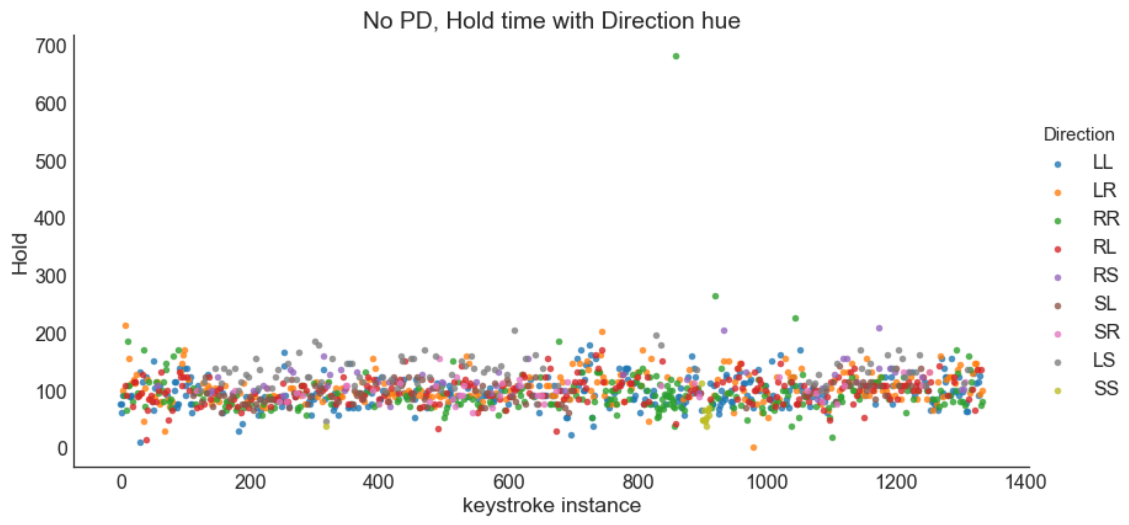


Figure 3 Exploratory Hold time plot for a user with no PD

As for hold time with hand hue:

Notice that there is no obvious trend whether a certain hand has advantages over the other, even in the case of PD sidedness.

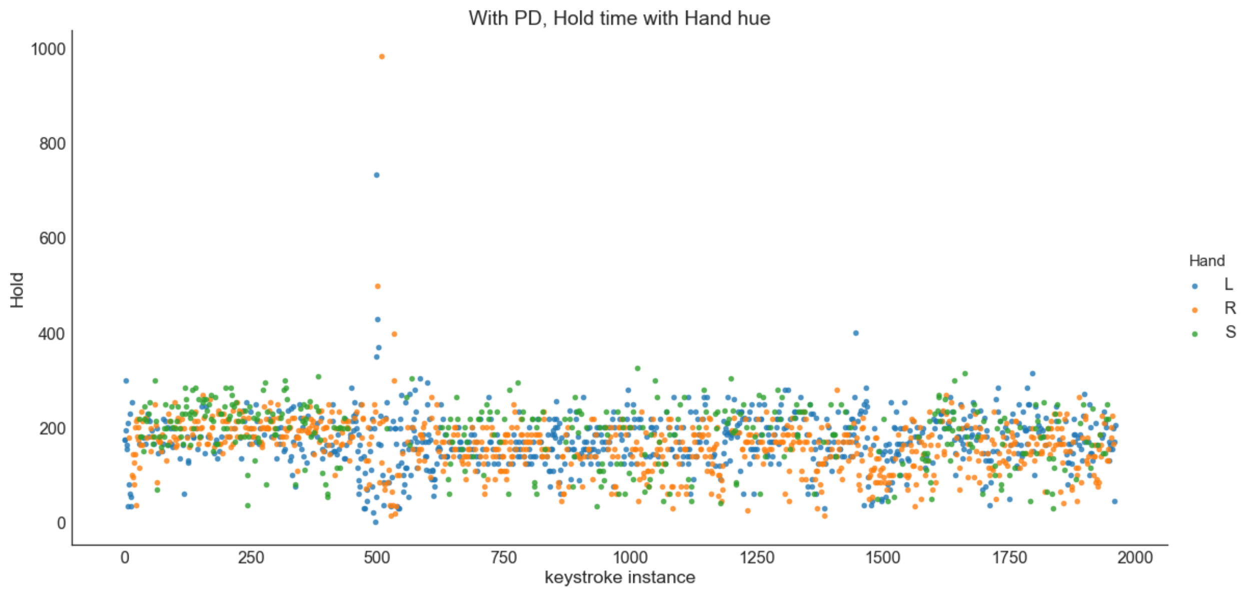


Figure 4 Exploratory Hold time plot for a user with PD

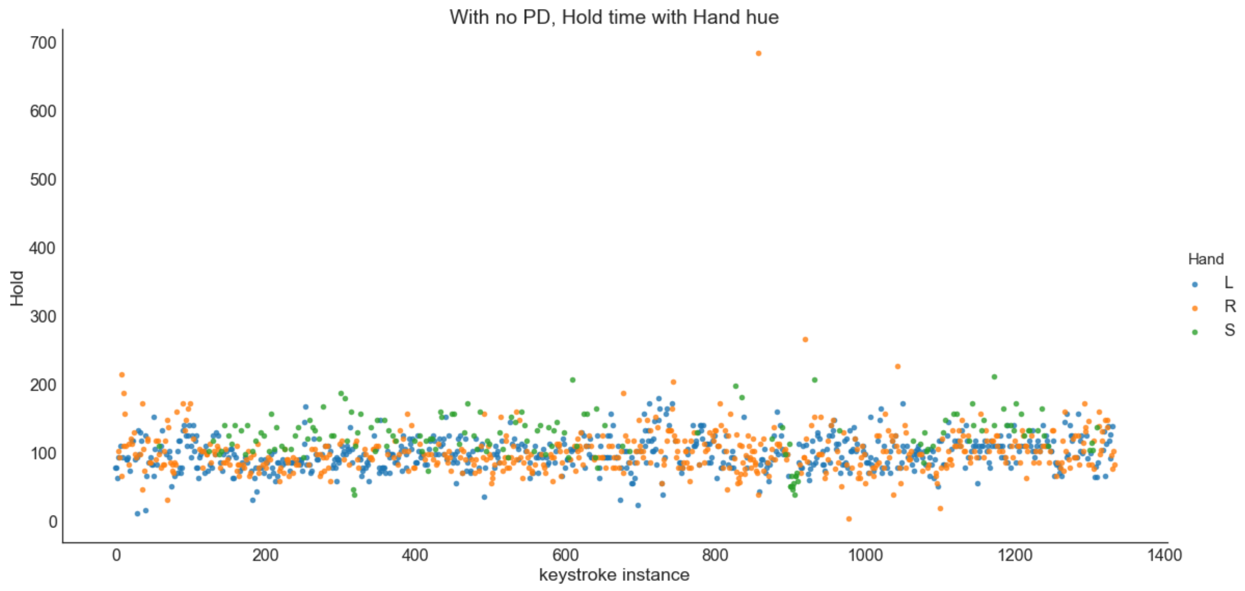


Figure 5 Exploratory Hold time plot for a user with no PD

As for flight time with hand hue:

Notice that there is no identifiable trend as well between PD and no PD group.

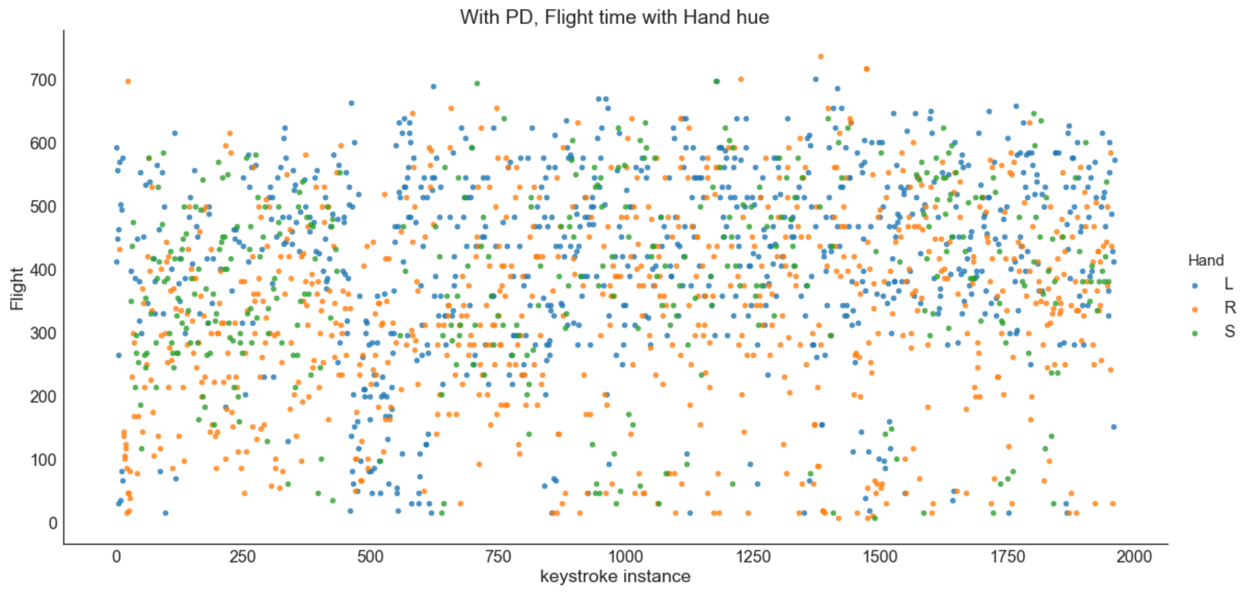


Figure 6 Exploratory Flight time plot for a user with PD

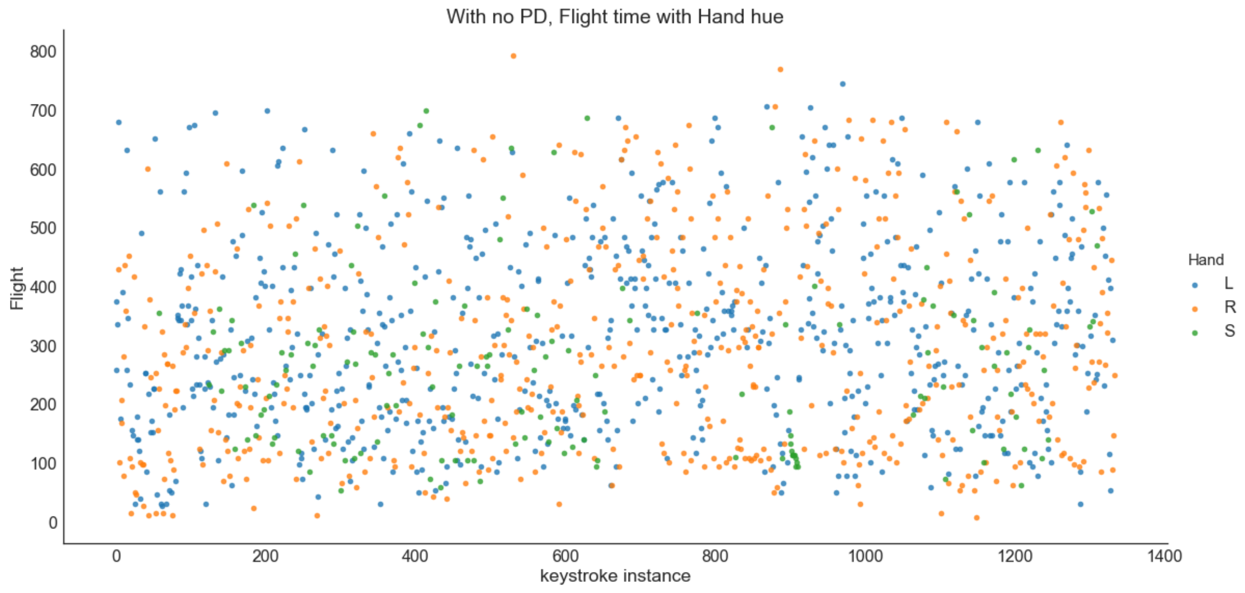


Figure 7 Exploratory Flight time plot for a user with no PD

Due to the attribute that these timing are so sparse, it was decided that some statistical means are needed to find some engineering features that could help modeling later. Variances, means, quartiles are tried and there is not much success in modeling. However, when a histogram is plotted, there is actually a normal distribution taking shape. Thus, binning was applied.

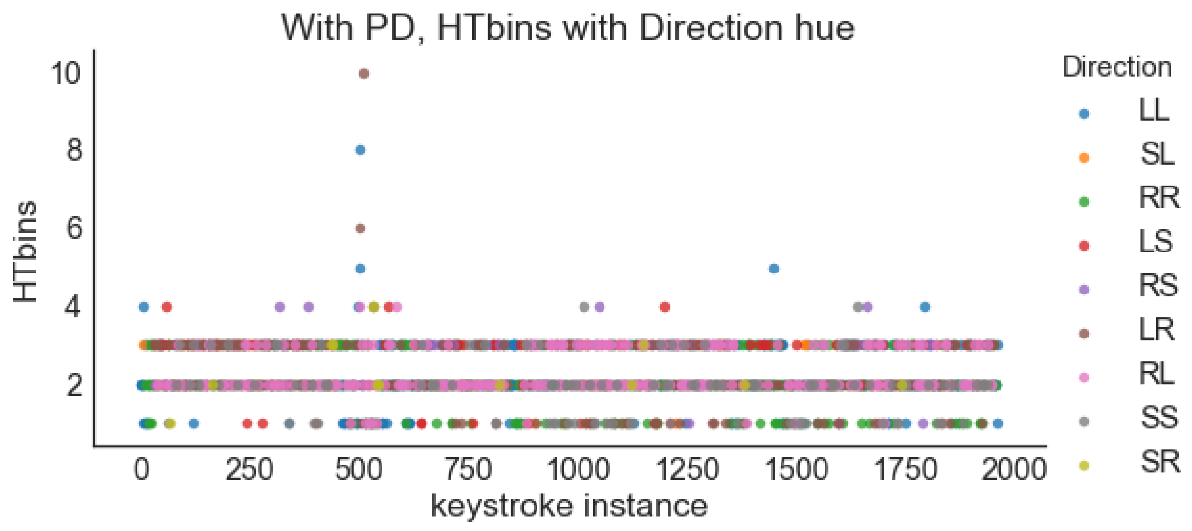


Figure 8 Exploratory HTbin plot for a user with PD

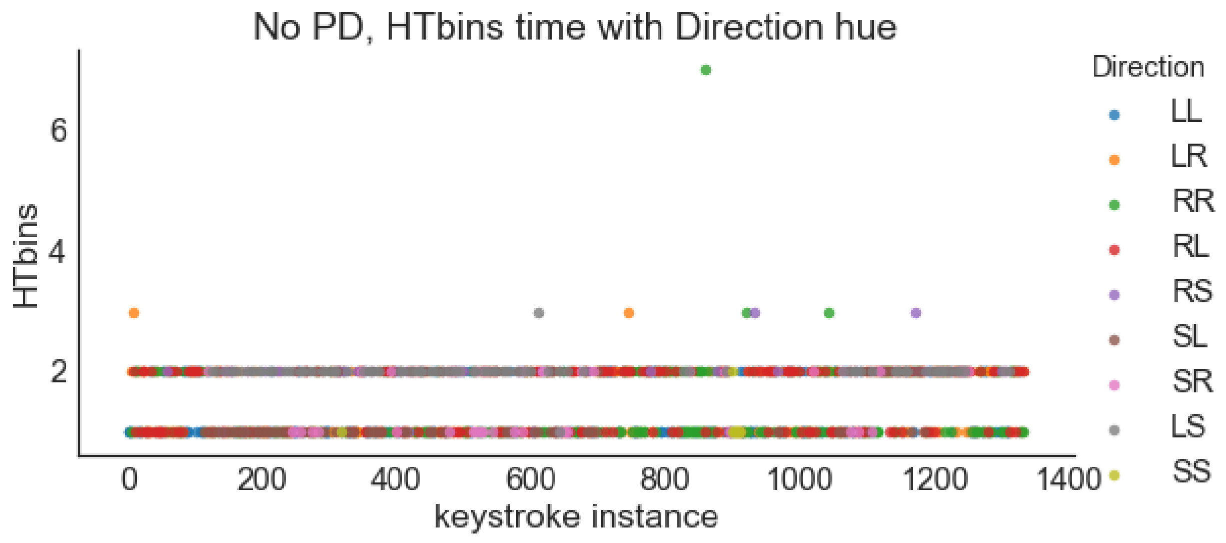


Figure 9 Exploratory HTbin plot for a user with no PD

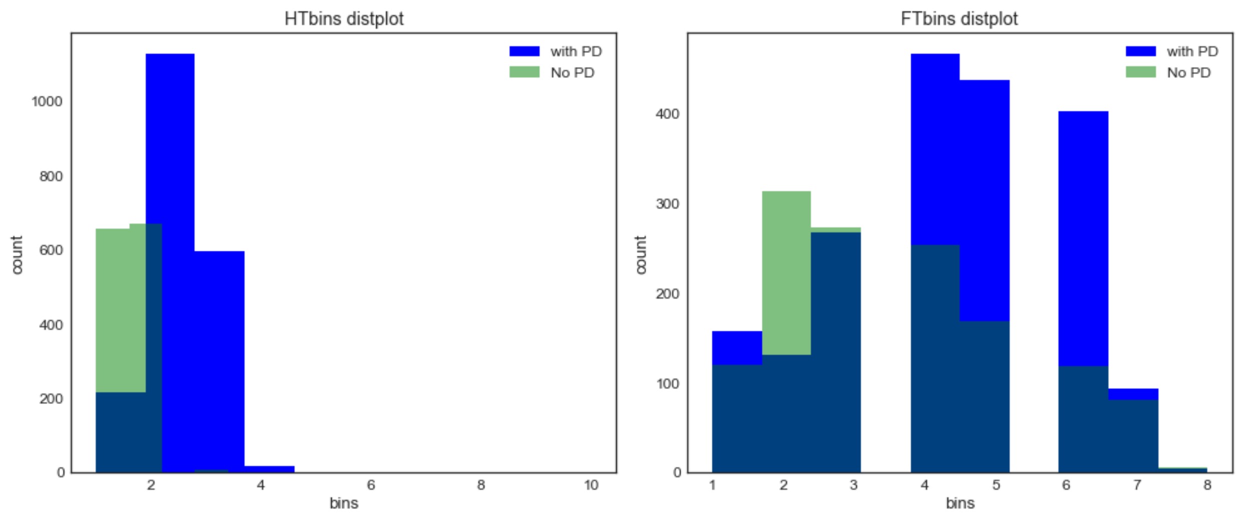


Figure 10 Histogram of HTbins and FTbins for all user separated by PD and no PD

Algorithms and Techniques

This is a binary classification problem that should focus on recall rate, with reasonable precision. Ideally all patients with Parkinson's disease should be detected, hence high recall rate for beneficial treatment and monitoring. This will invariably sacrifice some precision, causing inconveniences to falsely identified population that they are having Parkinson's disease. In this case, it is worthy since the inconvenience should be limited to monitoring and precautious action and not the actual treatment.

The algorithms used here are mainly ensemble machine learning techniques. There are

- Bagging algorithm:
  - bagging with decision tree classifier
  - random forest (essentially a bagging algorithm)
- boosting algorithm:
  - Gradient boosting
  - XG boosting
- Voting algorithm:
  - Combine best three of the above algorithms

Data are run through a few python functions to convert them from string form to float type when necessary, like for hold time and flight. Keystroke data are combined with User file to form a complete comma delimited file. Outliers are discarded. Hold times and flight times are separated into 10 bins each. Each bin carries 100ms, for a total of 1000ms to cover all duration needs. Variance and mean time for hold time and flight time are also calculated and written out to a csv file for python pandas to read in later on.

In NQ data set the timing is clock timing, meaning the keystroke in whole file is time stamps logging for all keys entered when activity get started. They went through the same process to arrive at the same state. Spyder 3 in Anaconda is used as coding environment.

Two huge files are produced and jupyter notebook was used to further process them as pandas DataFrame.

In jupyter notebook the 2 files are combined to form a DataFrame. It is checked for null values and those rows are discarded as there is no way to predict a replacement. No data is better than misleading data which could confuse models.

Bins contains the normalized frequencies of the assigned intervals. Adding all values for all ten bins will equal one. And a value of 0.3 in a hold time bin means that the bin contains 30% of all hold times of that user, and those hold times are all within the assigned interval for that bin. Bins that are logging zero for all users are removed.

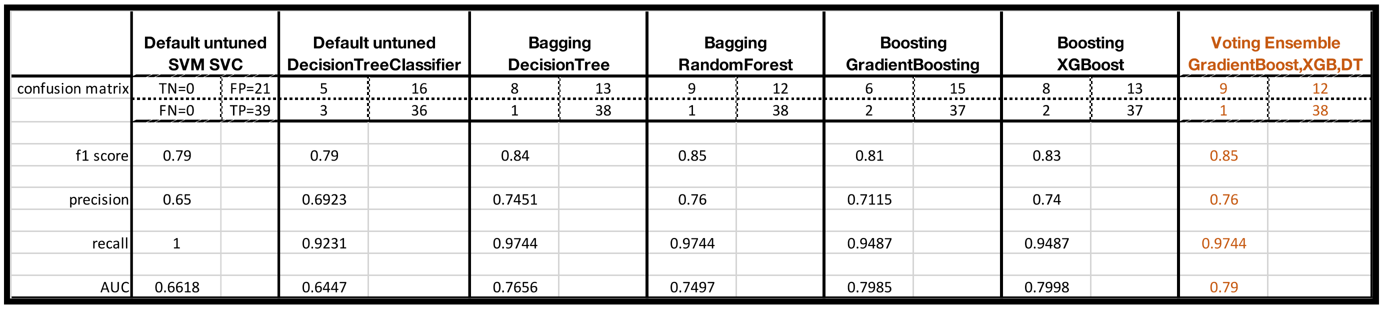
As for the variances and means they are run through models as is the first round. Then a MinMaxScaler from sklearn is used to transform them to between zero and one to facilitate the learning models. It was found that there is slight improvement.

20% of all data are reserved for testing strictly. Training data are used with cross validation technique. Kfold of various values and shuffling are used with the cross validation. GridSearchCV technique is applied as well to comb through parameters in search for optimum model.

A few common functions is extracted out. They are performance metric for GridSearchCV, and printing and plotting of results and ROC/AUC and PR curve.

Results

Model Evaluation and Validation



	Default untuned SVM SVC		**Default untuned DecisionTreeClassifier**		**Bagging DecisionTree**		**Bagging RandomForest**		**Boosting GradientBoosting**		**Boosting XGBoost**		**Voting Ensemble GradientBoost,XGB,DT**	
confusion matrix	TN=0	FP=21	5	16	8	13	9	12	6	15	8	13	9	12
	FN=0	TP=39	3	36	1	38	1	38	2	37	2	37	1	38
f1 score	0.79		0.79		0.84		0.85		0.81		0.83		0.85	
precision	0.65		0.6923		0.7451		0.76		0.7115		0.74		0.76	
recall	1		0.9231		0.9744		0.9744		0.9487		0.9487		0.9744	
AUC	0.6618		0.6447		0.7656		0.7497		0.7985		0.7998		0.79	

There are 60 rows reserved for testing. Among them, 21 are no PD, 39 are with PD.

Result of Voting Ensemble:

F1 score wise, only Bagging RandomForest and voting ensemble achieve 0.85.

As for recall, the best score is 0.9744, achieved by Bagging DT, Bagging RF and Voting Ensemble.

Among the Bagging RF and Voting Ensemble, they share the same high precision number. Thus, the AUC comes into play and the Voting Ensemble is a clear achiever, with 0.79.

In XGboost classifier, 4 features have been identified to be of utmost importance in differentiating the classes: FTbin2, FTbin3, FTbin4 and HTVar.

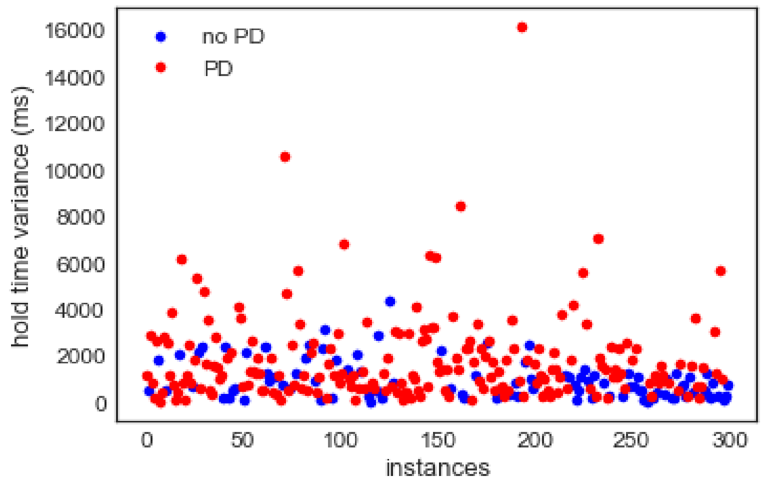


Figure 11 HTVar, one of the identified important feature

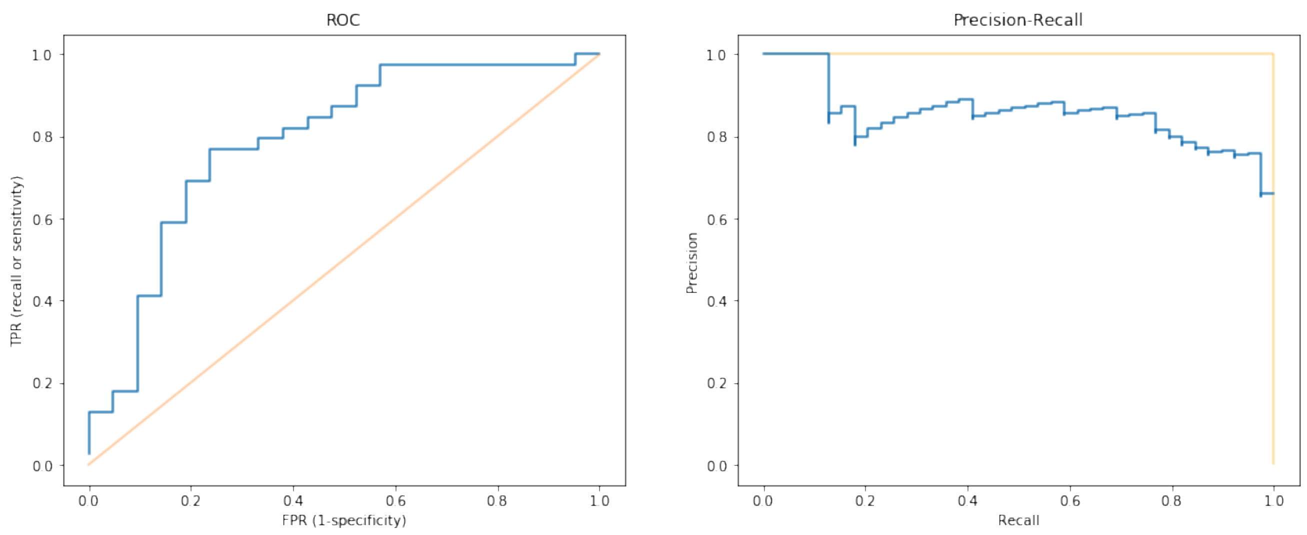


Figure 12 Result of Voting classifier

Justification

Recall of high percentage is mandatory here and precision of 0.76 seems to be a good enough sacrifice. Compared to benchmark model lightly tuned Decision Tree model, both the recall and especially the precision are improved.

The ROC curve of Voting Ensemble classifier is way better than the benchmark model.

Conclusion

Given the limited knowledge about the design of the two data collections and the attributes embedded within those keystroke data, the feature engineering is a challenge. Domain knowledge is a key factor as well in designing a successful machine learning model.

However, judging from the good PR curve and good ROC curve, together with the high recall percentage without totally thrashing precision, I believe a workable model has been achieved.

Voting ensemble borrows the strength of bagging algorithm and boosting algorithm. XG boost classifier has been flexible and highly effective and it truly contributes to the overall success.

Having said that, there are still ample room to improve. The parameter tuning can certainly be vastly improved. A deep learning model has been attempted and it just need a lot more effort to achieve its best potential.

The keystroke data can use some deeper insight and coupled with medical knowledge a lot more better features can be developed to hugely improve model without resorting to unnecessary complex modeling.

References

1. Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* **101**(23):e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/content/101/23/e215]; 2000 (June 13).

2. L. Giancardo, A. Sánchez-Ferro, T. Arroyo-Gallego, I. Butterworth, C. S. Mendoza, P. Montero, M. Matarazzo, J. A. Obeso, M. L. Gray, R. San José Estépar. Computer keyboard interaction as an indicator of early Parkinson's disease. Scientific Reports 6, 34468; doi: 10.1038/srep34468 (2016)

3. http://news.mit.edu/2015/typing-patterns-diagnose-early-onset-parkinsons-0401
Anne Trafton, MIT News Office
April 1, 2015

4. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5708704/
Holger Fröhlich
2017

5. http://xgboost.readthedocs.io/en/latest/python/python_api.html
6. https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/
7. https://www.kaggle.com/lct14558/imbalanced-data-why-you-should-not-use-roc-curve
8. https://acutecaretesting.org/en/articles/precision-recall-curves-what-are-they-and-how-are-they-used
9. https://jakevdp.github.io/PythonDataScienceHandbook/04.14-visualization-with-seaborn.html