Diagnosing Parkinson’s Disease using Voice: A Machine Learning Approach

By: Huy Phi

CSE 163 Project

1. Research Questions:

a. **Which voice features are most distinguishable comparing PD to healthy participants?** Will do statistical analysis including t-tests comparing features between PD and healthy groups, to try and filter down and select for features to input into machine learning models.

   Answer: Of the 755 original feature columns, 445 feature columns were found to have significant differences ($p\text{-value} < 0.05$) between the PD and healthy groups.

b. **Which machine learning method will be most accurate in diagnosing PD from voice?** I will explore different machine learning classification methods including decision tree, random forest, logistic regression, and k nearest neighbor from the Sklearn libraries. Voice features will be used to build these classification models, and classify PD vs. Healthy among the dataset. These models will be assessed for confusion matrix measures, including precision, recall, F1-score and accuracy, and cross-validated to get generalized (mean ± sd) measures for each model.

   Answer: Of the 4 models generated, the Random Forest model demonstrated the best performance. The worst model was the K Neighbors model.

c. **How can we further evaluate these machine learning methods?** I will use accuracy score for evaluation of accuracy of each model. Will also generate receiver operator characteristic (ROC) curves for the models to assess overall model performance to determine which model performs best as a binary classifier.

   Answer: Of the 4 models generated, the Random Forest model had the highest area under the curve (AUC) in the ROC curve. K Neighbors model had the lowest.

2. Motivation and Background:

Parkinson's Disease (PD) is the 2nd most common neurodegenerative disease, and affects primarily motor function with symptoms manifesting in muscle rigidity, bradykinesia (slowed/delayed movements) and tremor. With the continuing ageing of the population, the prevalence of PD will only continue to rise. Thus, diagnostic tools must be made more accessible to aid in early detection of PD. Voice is produced by over 70 muscles, and the symptoms of PD also include downstream acoustic effects on voice including changes in fundamental frequency, jitter, spectral power, etc. Previous studies have could diagnosis PD using extracted voice features and machine learning. Using voice has exciting potential as a diagnostic tool, because of the increasing accessibility to audio capturing devices, such as smartphones, smart speakers, etc. Optimizing voice diagnostic models can be used to increase accuracy, with the potential in early detection of PD across a larger subset of patients, leading to earlier access to treatment and better quality of life for patients with PD.

3. **Dataset**:

Link: https://archive.ics.uci.edu/ml/datasets/Parkinson%27s+Disease+Classification#

The dataset was pulled from the UCI Machine Learning Repository, and included extracted voice features from 188 patients with PD, and 64 age controlled healthy controls in the Department of Neurology at Istanbul University, Turkey. 754 columns are in the dataset, from speech signal processing algorithms are the features that will be used in machine learning. These feature sets include Time Frequency features, Mel Frequency Cepstral Coefficients (MFCC's), Wavelet Transform Based Features, Vocal Fold Features, and TWQT features, which can all be accessed in the link above.

4. **Methodology:**

Fortunately, the data has no missing values so the data did not need to be significantly clean. I read in the csv using pandas, and made sure to skip the first row, because it is empty. A column to drop is the tester id column, because participants were asked to submit at max 3 voice samples, so I want to eliminate any link of id between any training and testing in training our models. I then used seaborn barcharts to plot out the demographics of the dataset.

To answer question 1, I performed independent T-tests comparing PD vs healthy groups on all voice features. This will require use of pandas and SciPy libraries. I will then use an alpha value of 0.05, and drop all features with a p-value above that threshold from the data. Alpha value of 0.05 is often used as a cutoff for statistical significance, and essentially tells you that there will be a 5% chance that the relationship observed is due to chance, and not due to the difference between the two comparison groups. I then used seaborn boxplots to plot out the separation between the top 4 significant features.

For research question 2, I built classification models using SciKit Learn (Sklearn) library, and pandas. The predicted value will be for if the voice sample is diagnosed with Parkinson's or not, and the input features will be the significant features that we found. I used SKlearns Decision Tree Classifier, Random Forest Classifier, Logistic Regression, and K Nearest neighbor classifier packages to build the different models, with the label being class. Data was split into a training and test set of about 20% composing the test set, with a random state set for reproducibility of results. The only specific parameters I input in the models is for the RandomForest model, where we will set n_estimators (number of decision trees used) in between 10-100 depending on performance, and for the logistic regression model, where we will set the solver to "liblinear" to avoid the default solver warnings. These models will then be assessed for performance. To eliminate any convergence warnings that may come from the logistic regression, we set a warning ignore for Sklearn convergence warning.

Model assessment was done using a method called K-folds cross validation. Using the cross-validation score method from Sklearn, this will multiple train:test sets among the given data set, to get an average measurement of performance. Specifically, I will use 5-fold cross-validation. In addition to model accuracy, I calculated recall, precision and F1-score. A confusion matrix made to generate True/False positives (TP/FP), and True/False negatives (TN, FN). Recall is equal to TP/ (TP+FN), precision is equal to TP/ (TP+FP) and F1-Score is equal to (2 * Precision * Recall) / (Precision + Recall). Each model will be cross-validated for each of these scores (Recall, Precision, F1-Score, Accuracy). I then plot a sample confusion matrix from 1 testing set of data for a visual, using seaborn Heatmap.

To answer question 3, I generated ROC curves for the various models to evaluate how the models behave as we change the threshold of classification. An ROC curve essentially tells you how the model performs in classification of different threshold settings. Area under the curve (AUC) is used to assess general model performance, with the higher the better. This requires the ROC package from Sklearn, and matplotlib.pyplot. We will plot the true positive rate by the false positive rate for each model, generating ROC curves for each. We will then calculate an area under the curve (AUC) for each model, using Sklearns AUC method. Comparing AUC's for each model will further allow us to compare and assess model performance.

5. Results:

The dataset was distributed as demonstrated in the barchart in figure 1. This dataset had more participants with PD than Healthy, therefore it was an unbalanced dataset which should be considered later.

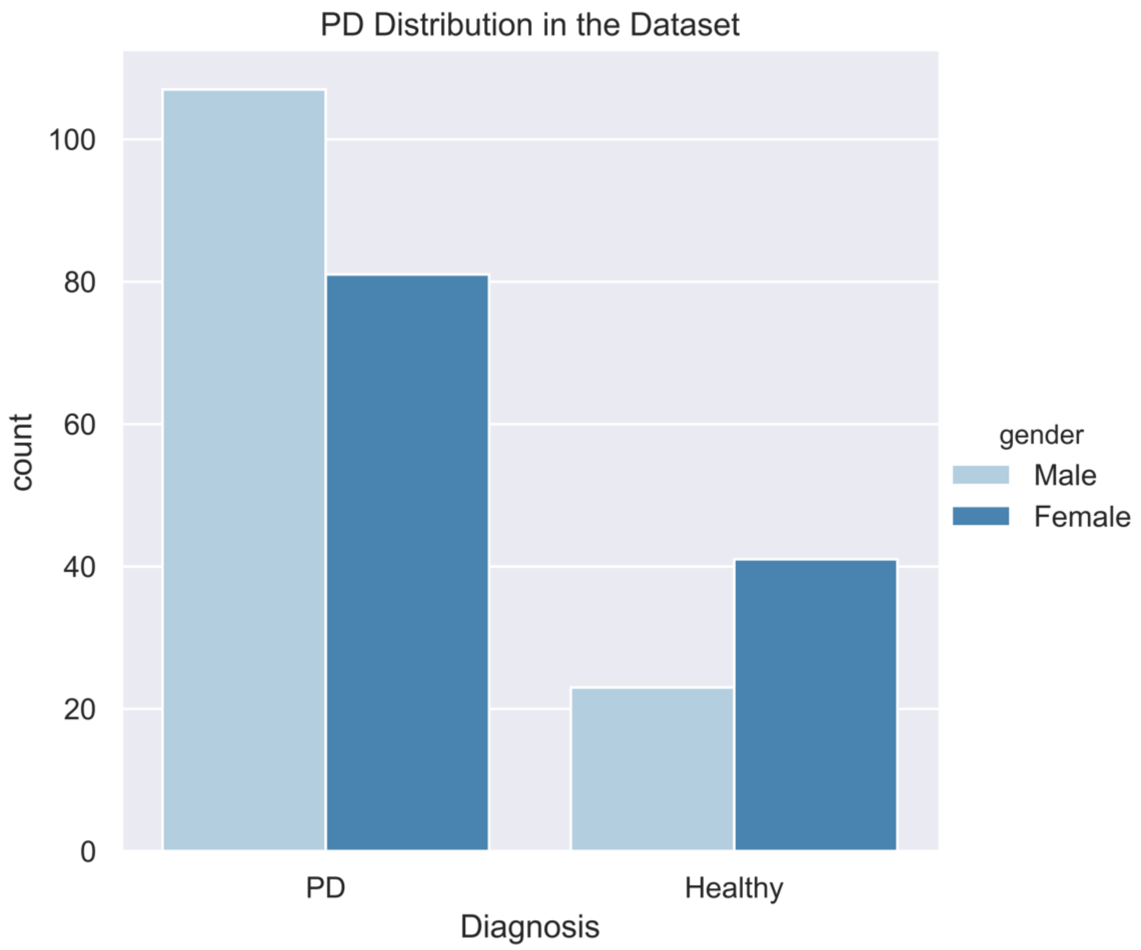

Figure 1. Bar chart showing the gender distribution among PD and Healthy participants.

a. **Which voice features are most distinguishable comparing PD to healthy participants?**

To answer this question, I performed independent T-tests on each feature in the dataset, comparing means of features between the PD and healthy groups using SciPy. I wrote functions that would generate dictionaries, mapping each feature to the p-value from the T-test, and then used a cutoff of a p-value < 0.05 to indicate statistical significant differences. I generated some tests in test.py to test my p-value calculations by hand. Overall, the dataset originally had 755 feature columns, but after setting a threshold cutoff of 0.05, the dataset was cut down to 445 feature columns. The top 10 features with the lowest p-values are shown in Table 1. Although the p-values were extremely low, my tests for the function demonstrated that they were calculated correctly. The most significant feature was the Mean MFCC 2nd Coefficient, but the TQWT feature set demonstrated tremendous promise, making up 8 of the top 10 features. Boxplots for the top 4 features were generated and is shown in Figure 2.

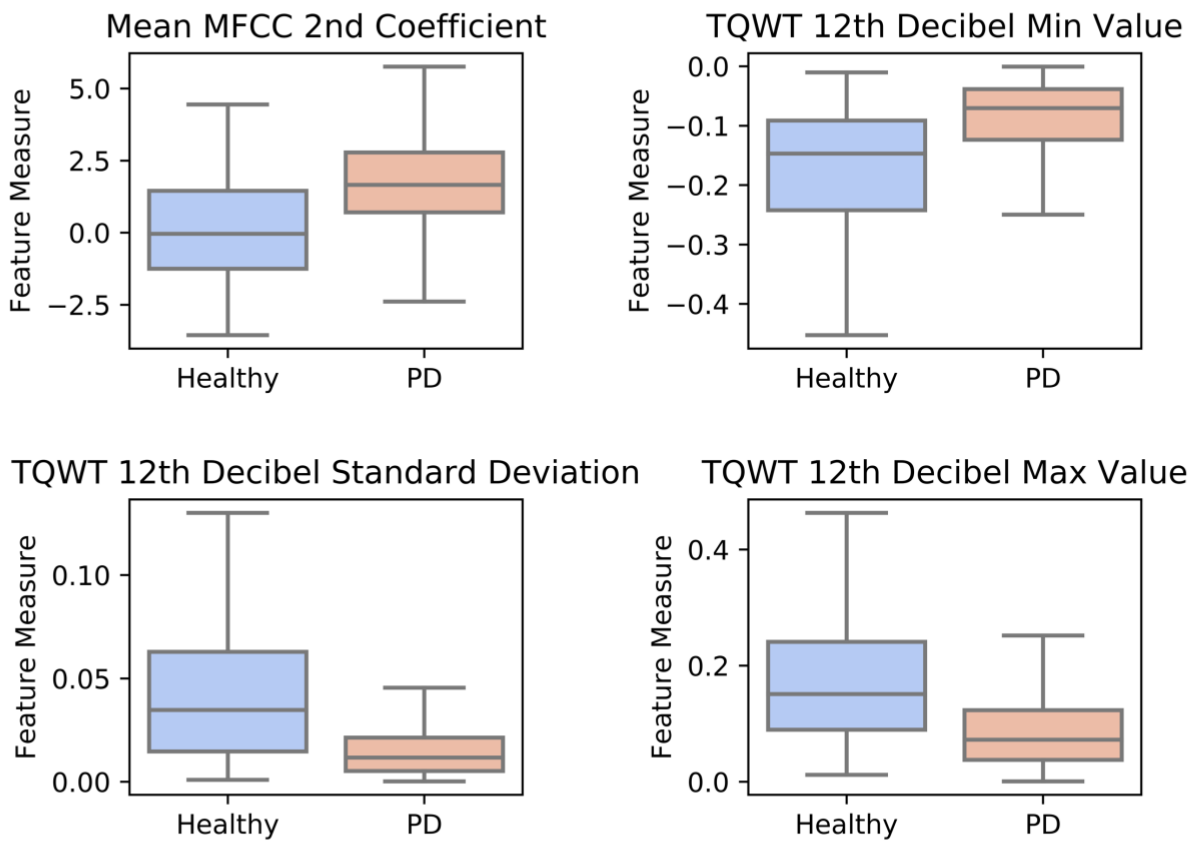

Figure 2. Boxplots of the top 4 significant features, comparing Healthy to PD participants.

Feature	**P-value**
Mean MFCC 2nd Coefficient	$3.28 * 10^{-30}$
TQWT 12th Decibel Min Value	$9.45 * 10^{-30}$
TQWT 12th Decibel Standard Deviation	$2.47 * 10^{-29}$
TQWT 12th Decibel Max Value	$4.56 * 10^{-29}$
TQWT 11th Decibel Standard Deviation	$4..90 * 10^{-29}$
TQWT 12th Decibel Entropy Log	$1.58 * 10^{-27}$
TQWT 11th Decibel Max Value	$3.70 * 10^{-27}$
TQWT 11th Decibel Min Value	$1.87 * 10^{-25}$
TQWT 13th Decibel Min Value	$2.83 * 10^{-25}$
9th Delta Delta Standard Deviation	$3.81 * 10^{-25}$

Table 1. The top features selected with the lowest p-values comparing PD vs. Healthy groups.

b. **Which machine learning method will perform the best in diagnosing PD from voice?**

To answer this question, I assessed each model using the K-Folds cross validation approach, to get a general sense of how each model performed. Using a fold value of 5 (i.e. generating 5 different train/testing sets) for each model, I found the mean and standard deviation scores for recall, precision, F1-Score and accuracy for each model, shown in table 2. Here we can see that in general performance, the Random Forest model scored the highest for all 4 measures. In terms of general accuracy, it was followed by the Logistic Regression Model, the Decision Tree model, and lastly the K-Neighbors model. Cross validation allows us to get a better picture of how a model will generally perform, and avoid overfitting/underfitting our models, due to only using 1 train:test set. Figure 2 shows a Heatmap demonstrating the confusion matrix for each model on a single testing set of data. We can see the random forest model getting the truest predictions compared to the other models.

Model	**Recall (Mean ± SD)**	**Precision (Mean ± SD)**	**F1-Score (Mean ± SD)**	**Accuracy (Mean ± SD)**
Decision Tree	0.85 ± 0.06	0.84 ± 0.03	0.84 ± 0.03	0.76 ± 0.05
Random Forest	**0.95 ± 0.02**	**0.86 ± 0.02**	**0.89 ± 0.02**	**0.84 ± 0.02**
Logistic Regression	0.96 ± 0.02	0.78 ± 0.01	0.86 ± 0.01	0.77 ± 0.01
K-Neighbors	0.87 ± 0.05	0.77 ± 0.01	0.82 ± 0.03	0.71 ± 0.04

Table 2. Model assessments from cross-validation at the point of writing this report. The random forest model performed the best, and is in bold.

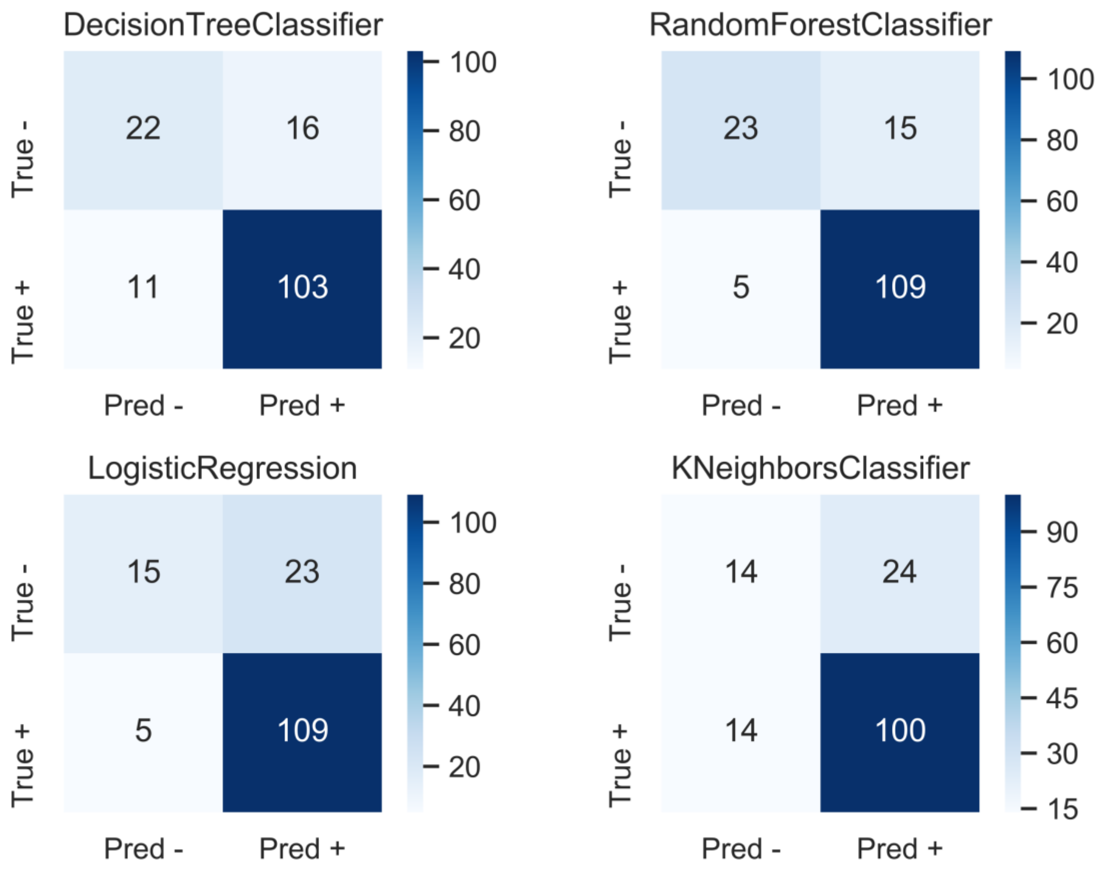

Figure 2. Heatmap of confusion matrix for each model on one set of test data.

c. **How can we further evaluate these machine learning methods?**

To answer this question, I calculated and generated ROC curves for each of the 4 models. The plot for ROC curves is shown in figure 3. ROC curves provide a way to assess general performance of a binary classifier, by plotting the false positive rate vs the true positive rate across different thresholds, and calculating the AUC. The closer an AUC is to 1.0 the better. Therefore, we see the highest AUC coming from the Random Forest model with and AUC of 0.996, followed by the Decision Tree (0.954), K-Neighbors (0.815) and Logistic Regression (0.792). This further validates that the Random Forest model is indeed the best model in diagnosing PD, but now shows that the Logistic Regression model was the worst.

The logistic regression model showed a high recall, and was good at predicting a positive value (i.e. if someone had PD), however this also came at the cost of its precision, where it also predicted a lot of false positives. Therefore, the logistic regression model demonstrated a lower AUC than the K-Neighbors, even though it had a higher accuracy. This is particularly important when developing a classifier is used in diagnosing disease, you want to limit the number of false positives generated by the model, to avoid any unnecessary grief, stress and costs associated with treatment. Thus, all of these models would want to be further expanded onto larger datasets and tinkered with to try to achieve higher accuracy.

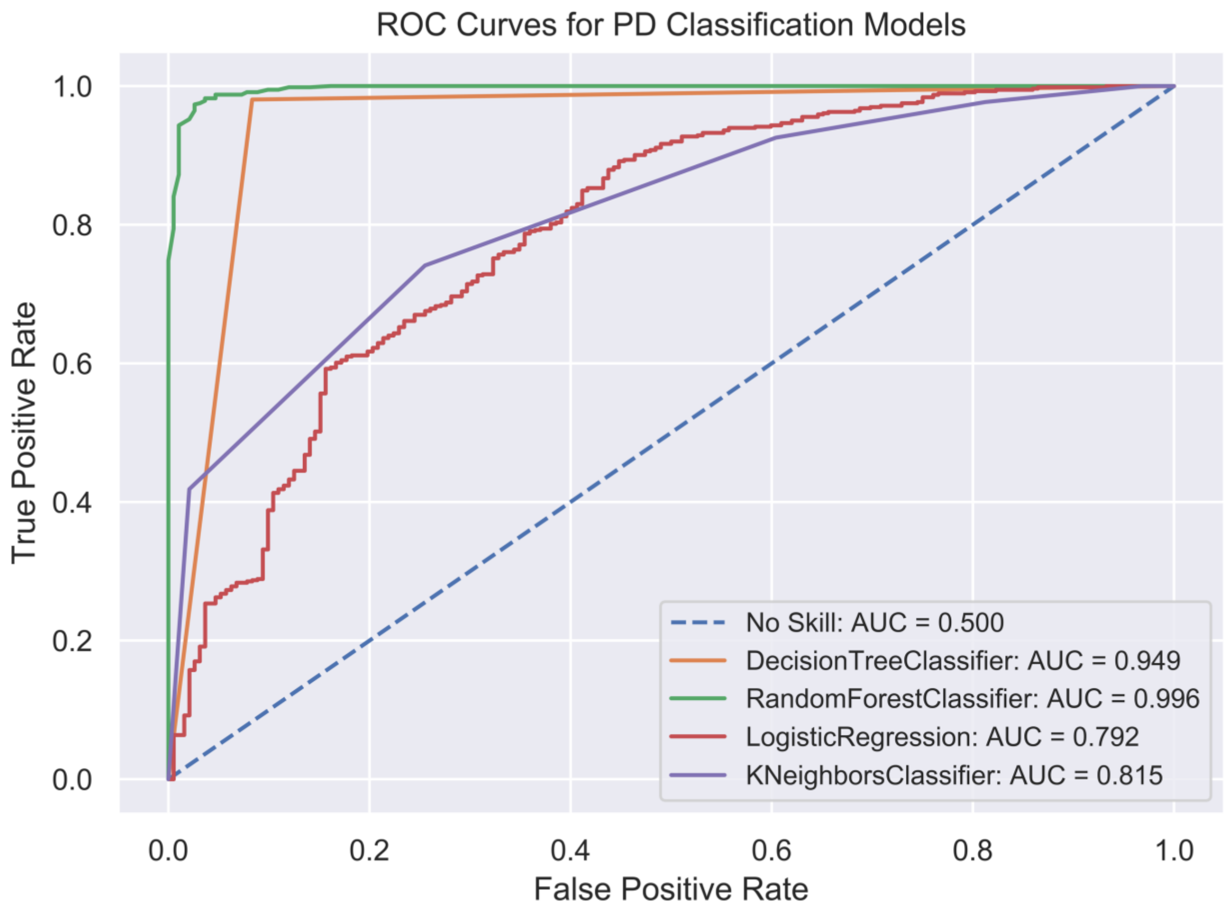

Figure 3. ROC curves for each model. We see the Random Forest model has the highest AUC, with the Logistic Regression model having the lowest.

6. <u>Challenge Goals:</u>

Machine learning:

My project completes this goal, because I built machine learning classification models with the inputs being the voice features to classify PD vs. healthy participants. I utilized different machine learning methods from the Sklearn library, including decision tree, random forest, logistic regression and K-neighbor's classifiers, and did all the necessary data processing necessary in Python. I also assessed the machine learning models I generated, by evaluating them via confusion matrix, and ROC curve.

Validity:

My project completes this goal because conducted statistical testing comparing feature differences between PD and healthy participants, to see that there is indeed significant difference between the two groups. These features were then used to filter out significant vs non-significant features. Additionally, I validated my machine learning models by doing cross-validation using the K-folds method, to get an average assessment of model performance. By dividing the data into multiple train-test sets, this ensured that the models were not overfitted/underfitted by a random train/test split and allowed for performance to be generalized, further validating the performance

7. Work Plan Evaluation:

Initial Work Plan:

Data set up and statistical testing: 5-8 hours

Set up in pandas will be quick, however I must become familiar using SciPy to generate T-tests comparing diseased vs non-diseased groups. I suspect this to take a long time to learn a new library. I understand the conceptual portion of this, but am learning the code for the first time.

Building our machine learning models: 3-5 hours

Building various machine learning models should be relatively simple, as I have already done so in this class. The learning curve will be when I choose to implement other models that we have not used in class, which will contribute mostly to the time learning how to implement this algorithm.

Generating Confusion Matrix and ROC curves: 8-12 hours

I expect this portion to take the longest amount of time, because I will be learning how to implement something I did not learn in this class. Although I understand conceptually these methods, doing the code is something new for me, and navigating through the Sklearn packages for ROC and confusion matrix will occupy the most amount of time.

Generating other plots for presentation: 1 hour
I plan to generate some plots of gender/disease distribution in the data set for the paper/presentation using seaborn which shouldn't take long to do.

Evaluation:

Data set up went well. The dataset entirely clean with no empty values. I just needed to set up my CSV by skipping the first row, which was an empty row, and was a quick fix with reading the necessary documentation. Setting up my statistical testing was challenging because I had to read up on SciPy documentation to learn how to utilize the t-test method. I also wanted to make sure my calculations for p-values were correct, so I generated a test file which I did not think I would need to do initially. Building my statistical functions took about 6 hours of time, so this was an accurate estimate of workload, especially with me not working with this library before.

In terms of building the machine learning models, I vastly overestimated the time frame, and this was done in a matter of 1 hour. Turns out machine learning is very easy to implement! One thing I will say was I did spend about 2 hours going into hyperparameter tuning to try to maximize model accuracy. I achieved this with the logistic regression model, where I used an L1 penalty and a liblinear solver, and this could achieve and AUC of about 0.96, which is much greater than the AUC I reported for logistic regression. However, these parameters meant that the model took about a 1 minute to fit, and with the k-folds method, cross validating would then take an absurd amount of time, which is why I decided to opt out of this method. My exploration into hyperparameter tuning took an additional 2 hours, thus my estimate was accurate, but because I added on hyperparameter tuning.

In generating my confusion matrix, I initially was just going to perform a confusion matrix on 1 train/test split, and had written a function to do so for each model. However, I gained some insight from Trinh (my TA) and decided to follow her recommendation to perform cross validation across my models. Thus, I had to read to read up on documentation for cross validation and write a new function, which added additional time. Overall, the confusion matrix took about 5 hours to write. ROC curves took the most time, and I had to read up on a lot of documentation/examples on how to effectively plot these curves and calculate AUC. This took about 8 hours to do, and I wanted to optimize the ROC plot function so that I wouldn't write a lot of redundant code. Thus, my estimation for confusion matrix/ROC curve was mostly accurate, and was good because it was a lot of code/concepts that I needed to learn, and can use.

Lastly in generating plots, I initially just thought I was going to do just a barchart of demographics, and this took about 1 hour so that was an accurate estimation. However, in doing my statistical testing/confusion matrices, I decided to generate the boxplots and heatmaps for these assessments, which added an additional 2 hours of work. Thus, my estimations were off, I just didn't have a finalized set of figures I wanted to make at the time.

Overall, my estimations were about right for this project. Developing on Jupyter notebook and then transferring into a cohesive .py script is also an additional challenge, and I had to convert from writing in a notebook cell format, to a function/main method format which took about 2 hours altogether.

8. **Testing:**

   To test my statistical calculations, I generated the test.py file, and handmade 2 testing csv files. I used assert_equals function from CSE163 utils to test the functions I made in main, and if they could calculate p-values I calculated by hand, and validate my calculations on the larger dataset. Cross-validation helped gain a better generalization of model performance, and limit the randomness of any particular train/test data split on the results.

9. **Collaboration:**

   I worked on this project completely by myself, with the help of my TA Trinh, and online resources including the course website, documentation for the libraries I used, as well as Stack Overflow for when I ran into problems.