AI-Assisted Clinical Decision Making for PD Treatment Selection

Jad Dibs

Design Project: Computer Science

Table of Contents

Acknowledgements

I would like to start by thanking my parents for supporting and encouraging me throughout this project.

I would also like to thank my sponsor and teacher as well as my Northwestern University mentor for guiding me whenever I sought help for my project.

Finally, I would like to acknowledge the developers of the Google Machine Learning Course with TensorFlow APIs. The knowledge I gained from the course assisted me with the development of the program in this study.

Review of Literature

Introduction

Today, Parkinson's disease (PD), progressive neurodegenerative disease, negatively impacts the lives of more than 10 million people worldwide (Parkinson's Foundation, n.d.). Unfortunately, medication errors and pharmaceutical industry influence are prevalent in treatment for PD, causing many patients with the disease to not be taken care of properly and to have prolonged or exacerbated symptoms.

The two aspects of medication errors that occur most often for people with Parkinson's disease during hospitalization are inaccurate prescribing of medicines and prescribing of pharmacologically inappropriate medicines. Inaccurate prescribing of medicines signifies incidences when a medicine is prescribed with incorrect dosage, incorrect timing, incorrect omission errors (medication incorrectly excluded from treatment administration), wrong formulation, wrong strength, etc. Prescribing of pharmacologically inappropriate medicines signifies incidences when a medicine is prescribed that conflicts with a patient's situation (contraindicated medicine or medication). Examples of such contraindications (considering medicines and surgical therapies) can include the following: patients taking medicines that oppose the effects of dopamine, which is integral for treatment of PD symptoms; patients with other medical conditions that may be worsened by certain medicines; patients having previous surgeries that place a risk on further surgical therapies; etc.

A study in Australia found that over 30% of all medications involved an inaccurate prescription incident during hospitalization, with the most common type of incident being incorrect timing (59.5%), followed by omission error (15%) and incorrect dosage (14.5%). Furthermore, it was determined that inappropriate medicines were prescribed to 15% of patients

with PD, with over half of these patients administered at least one dose (Bakker et al., 2012). Similarly, a journal from the US Pharmacy and Therapeutics Committee estimated that one in three patients with Parkinson's disease has been prescribed contraindicated drugs during hospitalization. Serious complications, mostly neuropsychiatric, have occurred in more than half of these patients (Grissinger, 2018). Overall, these medication errors are not only associated with increased length of hospital stays in PD patients, but also with a higher mortality rate (Lertxundi et al., 2016).

Type of incident	Prescribing errors	
	n	%
Incorrect dosage	25	14.5
Incorrect timing	103	59.5
Omission error	26	15
Wrong medication	4	2.3
Wrong formulation	8	4.6
Wrong strength	7	4

Figures 1-2: Statistics regarding prevalence of medication errors in PD treatment

The scale of the pharmaceutical industry also plays a large role in the mismanagement of treatment for patients with PD. Through marketing and enticement strategies, pharmaceutical companies significantly influence prescribing patterns by healthcare providers. Pharmaceutical industry and its sales representative's interactions and acceptance of gifts from the company's PSRs have been found to affect providers' prescribing behavior and are likely to contribute to irrational prescribing of the company's drug (Fickweiler et al., 2017). Moreover, the American Academy of Neurology reaffirmed that there are ample data showing that providers who take money from a pharmaceutical company are more likely to write prescriptions for that company's products, more likely to recommend that the company's products be added to hospital

formularies, and more likely to publish research findings favorable to that company (all of which can lead to ineffective prescriptions) (Elliott, 2014). The Ethiopian Journal of Health Science noted this inauspicious influence, with a review stating that the use of valid and reliable practice guidelines (that is, an alleviation of domineering pharmaceutical industry presence) could promote more rational and effective prescribing (Davari et al., 2018).

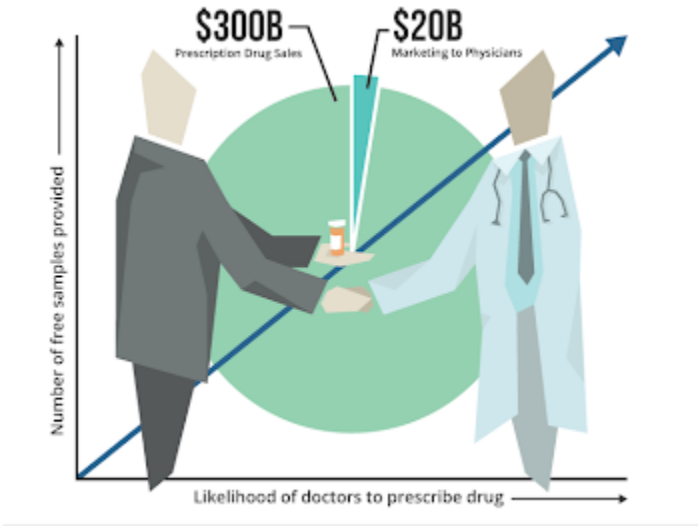


Figure 3: Graph showing general influence of pharmaceutical companies on physician (healthcare provider) prescriptions

Purpose

A machine learning model, however, can eradicate or at least produce a great reduction in the aforementioned issues in PD treatment: medication errors and pharmaceutical industry influence. By considering various factors of an individual patient's medical history, the model can assign an effective treatment plan void of contraindications, incorrect dosages, incorrect timing, etc. In addition, by being institutionalized as a requirement for healthcare providers to complete in order to aid their prescription decisions with unbiased input, this program has the potential to enhance the care of patients with Parkinson's disease worldwide.

Background Information

In order to create a machine learning model, there must be a large, high-quality dataset split into three subsets: training set, validation set, test set. Such a dataset should include various factors or data points relevant to the model. Splitting the dataset into three parts is a crucial part of the learning process. The training set establishes the foundation of the model, forming initial associations between provided data and factors. The validation set will refine the predictive abilities of the model by adding more data to form more associations. Finally, the test set will evaluate the predictive accuracy or other predictive metrics. For optimization, this process can be repeated. Developing the intrinsic model requires linear algebra and statistical techniques and can vary based on the creator; such techniques include support vector machines, random forests, neural networks, etc., all of which compile the data for classification and regression to predict values (Pruneski et al., 2022).

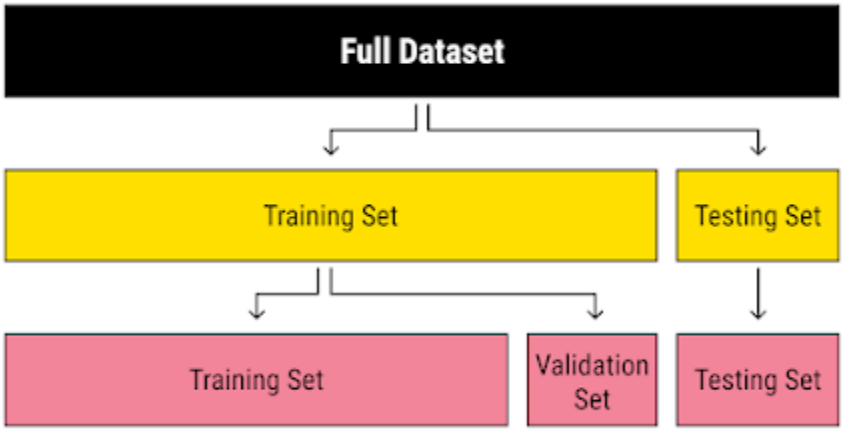


Figure 4: Splitting of dataset for machine learning model

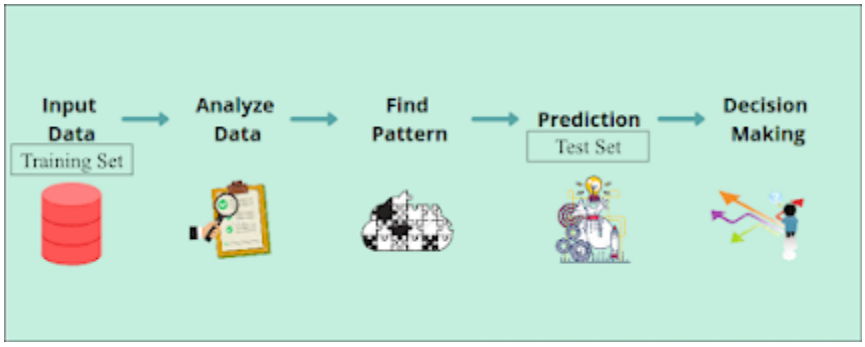
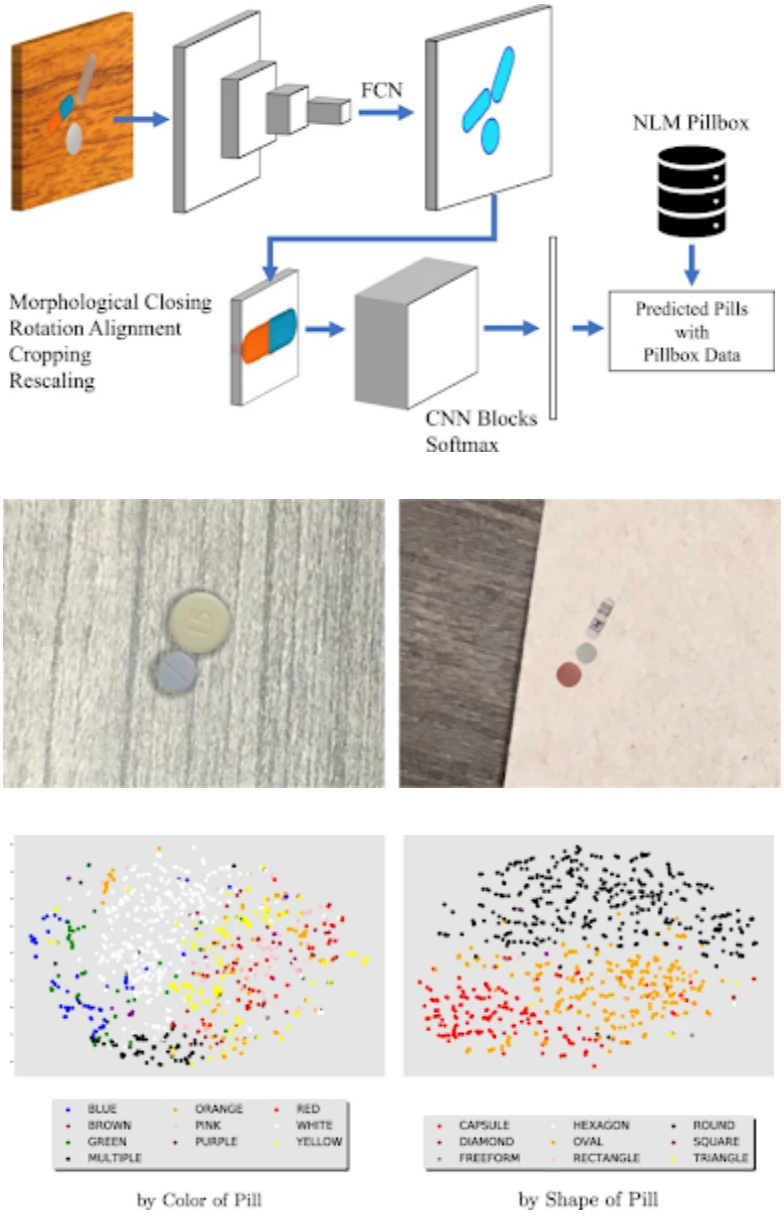


Figure 5: General process of machine learning model

To the knowledge of this study, no programs have been previously made to assign treatments for patients with diseases, particularly PD, using a machine learning model. However, programs have been made that have attempted to generally tackle two of the problems addressed: incorrect medications and incorrect medicine dosages.

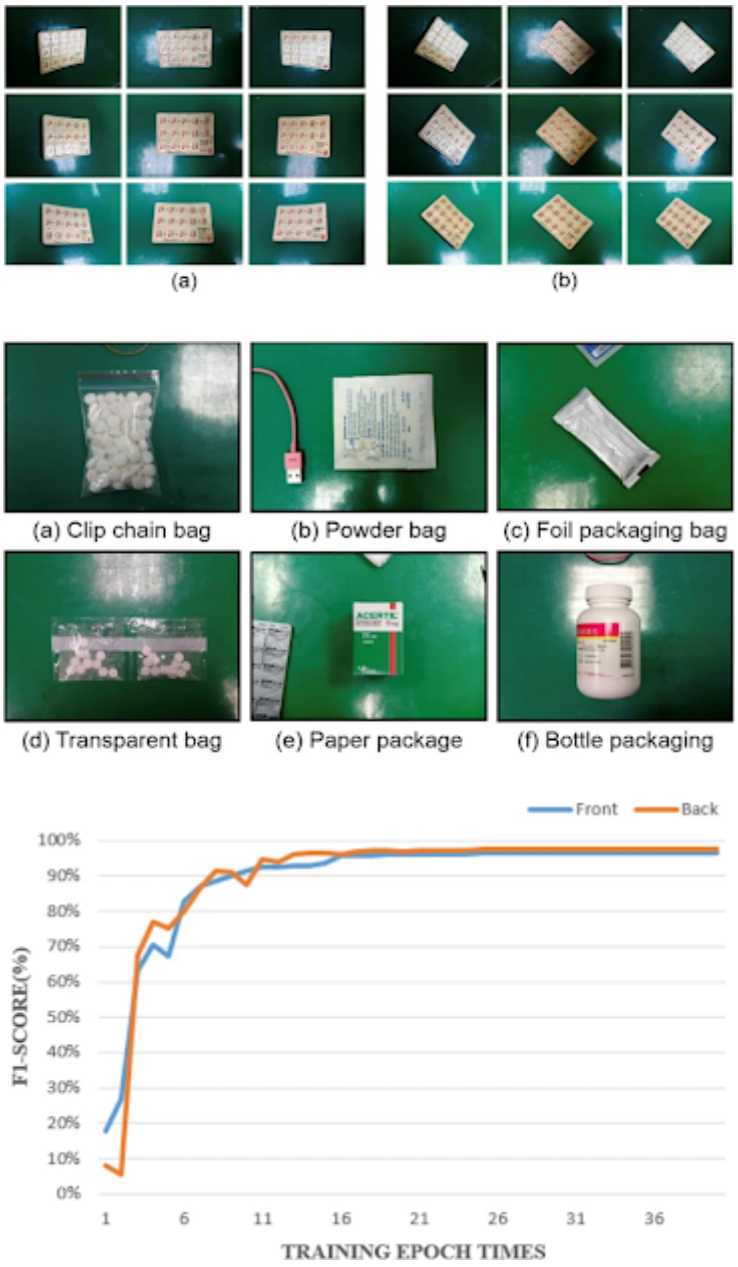
The programs that reduce incorrect medications implement a drug identification model, in which pictures of unknown medications are processed by a convolutional neural network (CNN), a type of machine learning model, to be identified. The primary aim of such programs is to eliminate the "look-alike and sound-alike" factor of medication errors, which occurs when the wrong medication is administered due to appearing to be the correct medication.

One of the first programs of this type was developed in 2019, utilizing a pill identification dataset provided by the US National Library of Medicine. In order to build the identification system, a pixel-segmentation model was used to localize the pills. Then, synthetic images were compared to the real pill images in the dataset; by training the model in this way, an association between the real medication images in the dataset and their medication names was formed, creating a program that identifies drug medications from images. This program was measured to have 94% accuracy (Delgado et al., 2019).



Figures 6-8: Images describing process of model in 2019 US study, showing CNN, synthetic images, and classification

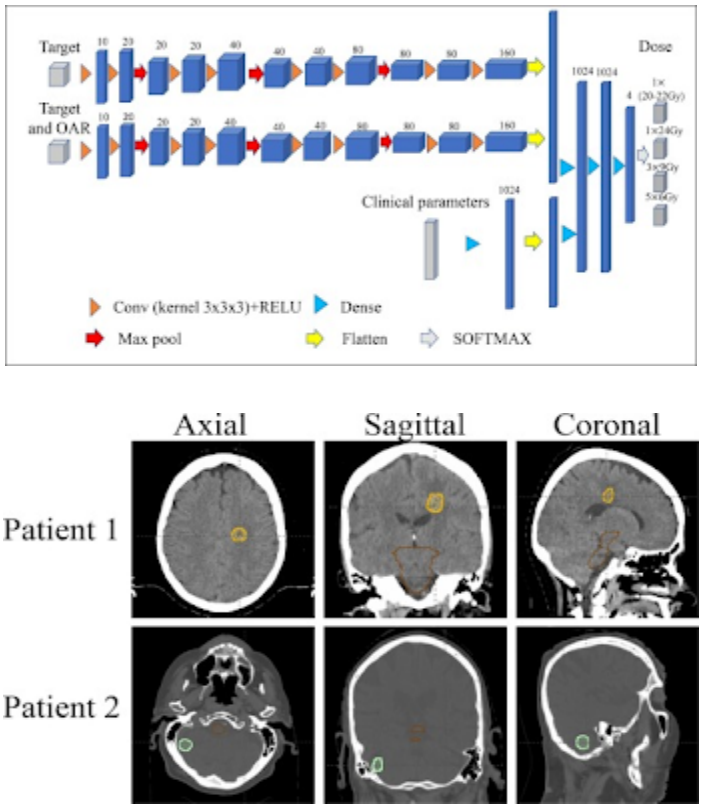
A similar technology was developed in Taiwan in 2020 with more than 90% accuracy. Although this program used a smaller set of drugs for the training set due to the absence of an established pill identification dataset, it compensated by incorporating all perspectives of a pill image in the model (front, back, side) (Ting et al., 2020).



Figures 9-11: Images describing process of model in 2020 Taiwan study, showing example dataset images and a performance metric (F1 score) graph

One program was found to reduce incorrect prescription dosages and utilized similar methods to the above studies. Created in 2023 and published in the Clinical and Translational Radiation Oncology Journal, this program furthered the field of AI-assisted clinical decision making by assigning dose prescriptions in radiosurgery of brain metastases. To create the

training dataset, structures of brain tumors were imaged on CT scans and processed into values (volume and distance to brainstem). Based on these data points along with information from the patient, prescription dosages were predicted with relatively high accuracy (82% accuracy) using a machine learning model (Cao et al., 2023).

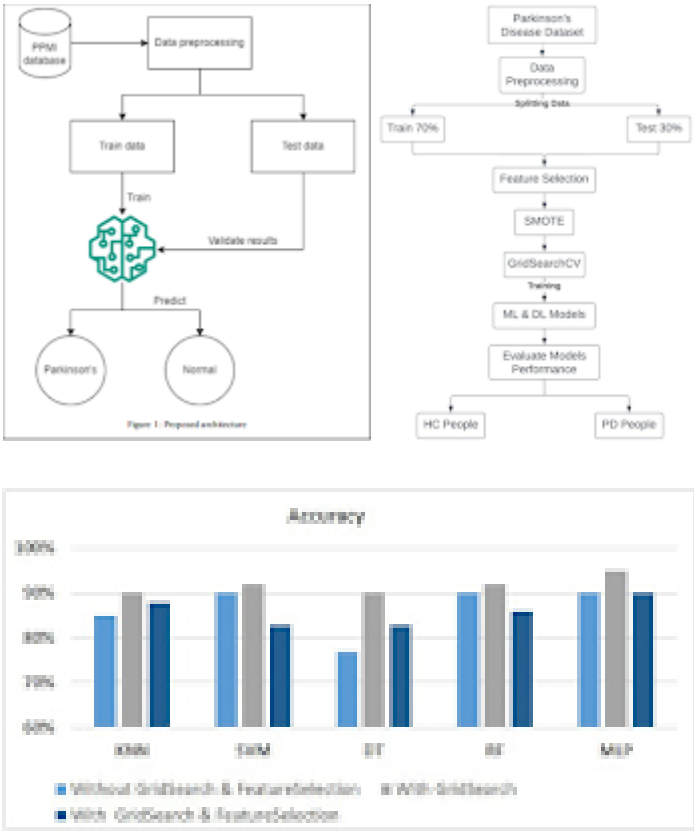


Figures 12-13: Images describing process of model in 2023 study, showing CNN architecture and example dataset images

A related topic in AI-assisted clinical decision making that is more relevant to the PD aspect of this study is diagnosis of PD or its severity using machine learning models. A correlation between phonetic features and early onset of PD was established. As a result, models for early diagnosis of PD began in 2020 and utilized voice detection. The distinction in extracted speech signals of PD patients and healthy people formed the training dataset; later, when speech

signals of those using the program were processed, they were compared to the dataset to determine the PD diagnosis (Senturk, 2020).

This field gained a reinvigoration in 2023, with two studies incorporating slightly varied methods for more precise speech signal processing (Alshammri et al., 2023; Govindu, Palwe, 2023). Another AI model was developed to remotely assess the severity of PD in patients by measuring motor performance with video recordings (Islam et al., 2023).



Figures 14-16: Images describing processes of models in aforementioned PD diagnosis studies, showing architecture of models and graph of performance metric (predictive accuracy)

While the studies mentioned above do share certain similarities with the intentions of the program in this study, such as reducing medication errors or involving PD, they do not directly relate to the purpose or context of this study. For instance, the programs that tackle the incorrect medications problem only consider the “look-alike and sound-alike” factor, which, while

important, does not cover all medication errors and only includes those in treatment administration by healthcare providers. The program in this study, on the other hand, seeks to reduce the prevalence of pharmacologically inappropriate prescriptions (contraindications) or incorrect prescription timing from healthcare providers, which occur prior to administration. The AI-assisted clinical decision making program that assigns prescription dosages is closely linked to one purpose of this study, but is for patients with brain metastases rather than those with PD. Also, these programs utilize images to draw conclusions, which is not a definitive factor in treatment planning for patients with PD and will thus not be used. Finally, the programs that diagnose PD and its severity are somewhat connected in that they involve an aspect of PD assisted by machine learning, but they differ in purpose with the program in this study (to prescribe treatments for PD). Ultimately, due to the novel nature of the program in this study and machine learning in general, there is minimal precedence to directly compare.

Design Description

The model in this study will utilize deep learning with neural networks, a form of machine learning called an Artificial Neural Network (ANN). Neural networks use machine learning to make decisions in a manner similar to the human brain, by using processes that mimic the way biological neurons work together to identify phenomena, weigh options and arrive at conclusions. Every neural network consists of layers of nodes, or artificial neurons: an input or feature layer, one or more hidden layers, and an output or label layer. Each node connects to others, and has its own associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

Neural networks rely on training data to learn and improve their accuracy over time. They have the capability to classify data quickly (IBM, n.d.).

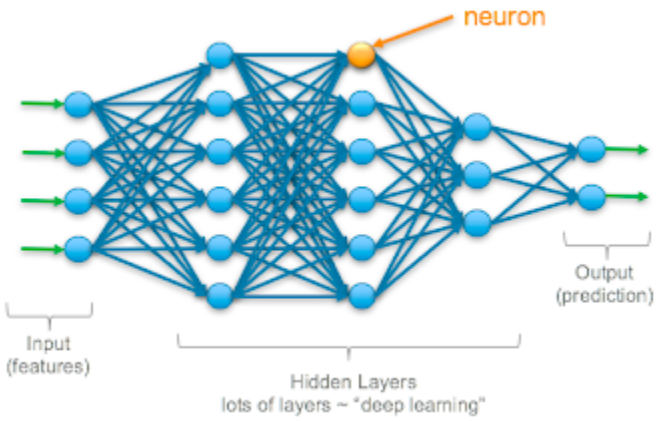


Figure 17: Example architecture of ANN, with input layer, hidden layers, and output layer

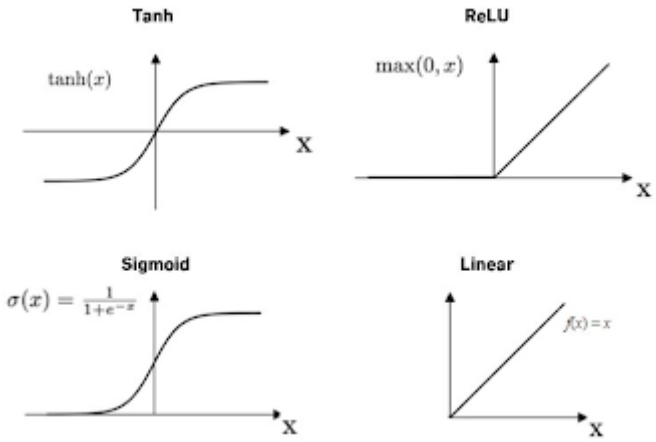


Figure 18: Various function options for determining classification threshold of layers

Specifically, the program will be a Multi-Label Classification Artificial Neural Network or Deep Machine Learning Model. The multi-label classification aspect means that the ANN can predict more than one output. Since the program aims to facilitate PD treatment prescriptions, the output will be the optimal treatment(s) (can be more than one simultaneously, a combination plan) for the patient with PD.

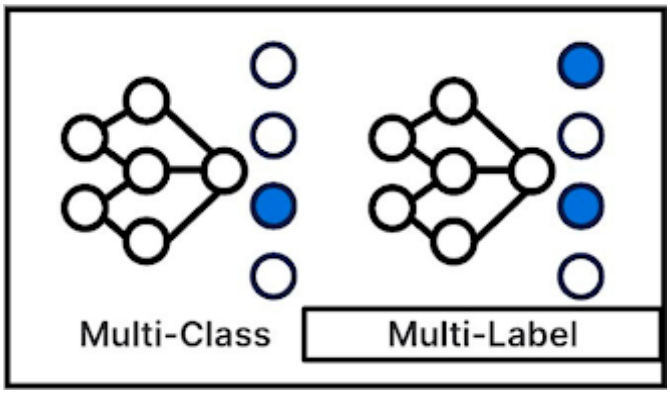


Figure 19: Multi-label allows more than one output, multi-class allows only one output

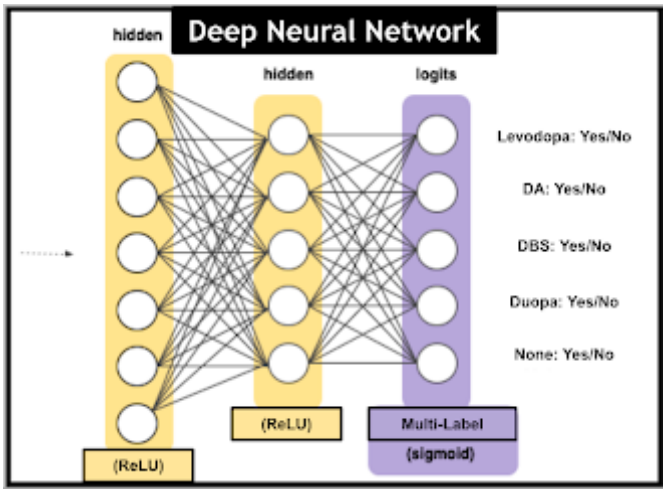
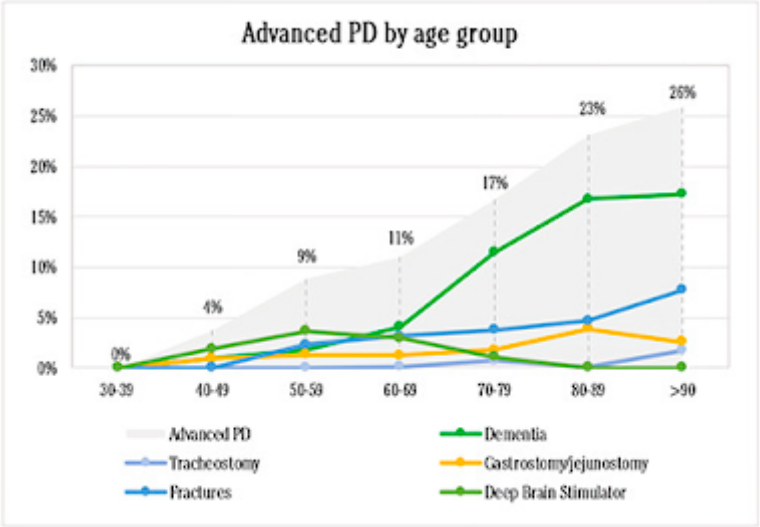


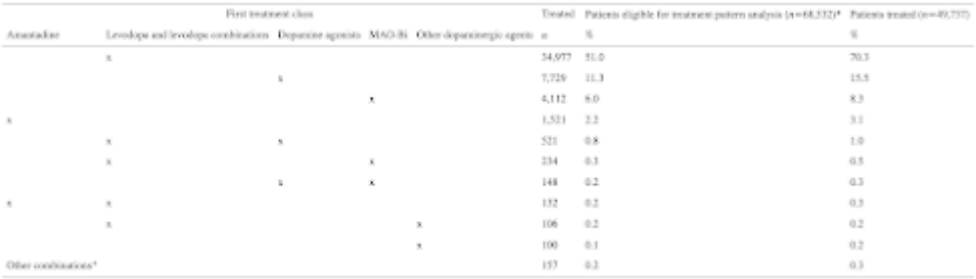
Figure 20: Architecture of ANN model in this study (potential outputs discussed later), excluding input layer

Conventional treatments for PD symptoms primarily come in two forms: medication or surgical therapy. These treatments are the potential outputs, or classes, of the ANN model. Various factors affect which treatment should be prescribed for ideal results, especially age, sex, stage of PD (associated with symptoms), and presence or absence of comorbid or medication contraindications (Parkinson's Foundation, n.d.). For a machine learning model that seeks to assign treatment(s) based on various factors, there must be a training dataset with data points that cover each aforementioned factor. Thus, ideally, the program in this study would collect and use

data of a large sample size of patients with PD provided by a hospital to form the training dataset of the machine learning model. However, due to an absence of open-source or accessible data regarding these specific factors, the program in this study is a simulation. Therefore, the dataset is made up of a manufactured, yet realistic, group of patients given differing medical backgrounds. Guided by numerous studies on treatment patterns and statistics of patients with PD worldwide, the data of and assigned treatment(s) for these patients reflect the general association between factors and treatment prescriptions in the real world in order to generate the most accurate simulation (Orozco et al., 2020; Willis et al., 2022; Conti et al., 2022; Rizek et al., 2016; Houghton et al., 2019; Medical Advisory Secretariat, 2005). Unfortunately, not all associations could be found or established due to absence of public data, which is discussed in the constraints. In the future, if a model is implemented in hospitals, the real data of patients will be inputted instead of this manufactured data.

Age (2015)		**Prevalence estimates per 100.000**	
	Global	**Male**	**Female**
30-39	2.9 (95% CI 1.8-4.45)	3.9 (95% CI 2.14-6.56)	1.9 (95% CI 0.73-3.96)
40-49	19.5 (95% CI 15.99-23.63)	25.7 (95% CI 19.98-32.51)	13.5 (95% CI 9.51-18.62)
50-59	91.6 (95% CI 82.71-101.29)	115.0 (95% CI 100.66-130.85)	70.1 (95% CI 59.46-82.16)
60-69	336.8 (95% CI 313.57-361.28)	408.8 (95% CI 371.70-448.61)	272.9 (95% CI 244.48-303.78)
70-79	1081.9 (95% CI 1019.27-1147.38)	1303.9 (95% CI 1200.95-1413.27)	906.5 (95% CI 830.36-987.84)
80-89	1786.1 (95% CI 1660.05-1919.19)	2236.9 (95% CI 2010.26-2482.16)	1508.6 (95% CI 1362.42-1666.26)
90+	1579.7 (95% CI 1305.37-1894.74)	1747.4 (95% CI 1269.68-2345.83)	1492.2 (95% CI 1167.58-1879.21)
All ages	157.92 (95% CI 152.25-163.43)	175.00 (95% CI 166.69-183.40)	141.91 (95% CI 134.85-149.24)





Figures 21-23: Example statistics used in the development of associations for the simulation dataset

The specific conventional treatments considered in the deep machine learning model in this study are the following:

- Prescription Medications (Parkinson's Foundation, n.d.)
    - Levodopa (Carbidopa-Levodopa) (Parkinson's Foundation, n.d.)
        - Levodopa works by being converted into dopamine in the brain. It is often viewed as the first-line drug for the management of Parkinson's motor symptoms, but it can produce severe side effects. Levodopa is almost always prescribed in combination with the drug carbidopa, which reduces or prevents the nausea that levodopa alone can cause. Levodopa is used in all stages of PD. At more advanced stages, however, surgery is usually advised instead. Contraindicated medications include narcotics/analgesics,

antidepressants, muscles relaxants, cough suppressants, decongestants/stimulants, monoamine oxidase inhibitors (MAOIs), antipsychotics, antiemetics, blood pressure medicines, and drugs to treat hyperkinetic movements.

- Dopamine Agonists (DA) (apomorphine, rotigotine, pramipexole, ropinirole, etc.) (Parkinson's Foundation, n.d.)
  - Dopamine agonists stimulate the parts of the brain influenced by dopamine. Since dopamine agonists mimic the effects of dopamine in the brain rather than convert into dopamine, they are less potent but produce less severe side effects. As a result, they often can only manage symptoms in the first few years and early stages of PD onset. Contraindicated medications are the same as levodopa.

- Surgical Therapies/Surgeries (Parkinson's Foundation, n.d.)
  - Deep Brain Stimulation (DBS)
    - Deep brain stimulation is a surgical therapy used to treat certain aspects of PD through an electrical pulse generator that stimulates the brain. This powerful therapy most addresses the movement symptoms of PD and certain side effects caused by medications. DBS is generally used in more advanced stages of PD. Contraindications are minimal.
  - Duopa (Parkinson's Foundation, n.d.)
    - Duopa is a therapy that directly delivers carbidopa-levodopa into the intestine in gel form rather than a pill through a tube implantation surgery. It is designed to improve absorption and reduce "off" times (changes in

movement abilities as a levodopa dose wanes). It is generally used for treatment in advanced stages of PD. Contraindicated medications are the same as levodopa.

Performance Criteria

As with all machine learning models, the primary quantitative indication of success or accuracy of the program is through performance metrics of a test set. In this study, the test set is a smaller dataset of a manufactured group of patients with PD without the treatment plan factor of information (Pruneski et al., 2022). Once this set is exposed to the model, the predictive machine learning tool will produce an output: the treatment plan that should be implemented. Since the program involves discrete outputs (each potential treatment being chosen or not), the way that the outputs can be measured using values and percentages is with a classification performance metric. Such metrics include the following: accuracy, binary accuracy, confusion matrix, precision, recall/sensitivity/hit-rate, precision-recall tradeoff, F1 score, and AUROC (Area under Receiver operating characteristics curve) (Bajaj, 2023).

Accuracy - Classification accuracy is defined as the number of correct predictions divided by the total number of predictions made by the machine learning model, multiplied by 100 (Jayaswal, 2020). Binary accuracy is the same as accuracy, but calculated for each individual class instead. The average of binary accuracies is another indication of overall predictive power.

Confusion Matrix - This matrix is a tabular visualization of the ground-truth labels (actual values in dataset) versus model predictions (predicted values in data set). The possible evaluation factors are true positive (actual and predicted are the same and are the “on” part of discrete classification), true negative (actual and predicted are the same and are the “off” part of

discrete classification), false positive (actual and predicted are different and predicted is “on” part of discrete classification), and false negative (actual and predicted are different and predicted is “off” part of discrete classification). Each row of the confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class. Confusion Matrix is not exactly a quantitative performance metric but can be a basis on which other metrics evaluate results (Jayaswal, 2020; Datatron, 2019).

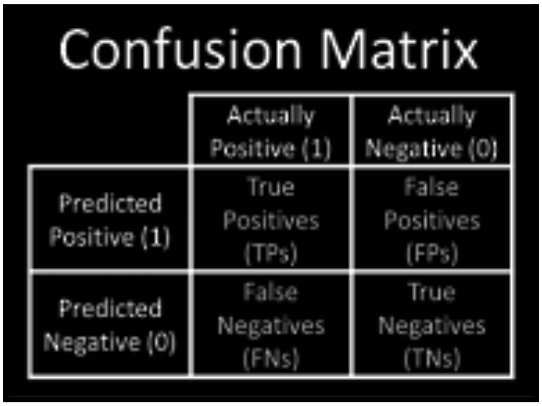
Precision - This metric is the ratio of true positives and total positives predicted (patients correctly identified / (patients correctly identified + incorrectly labeled patients)) (Bajaj, 2023).

Recall/Sensitivity/Hit-Rate - This metric is the ratio of true positives to all the positives in ground truth (Bajaj, 2023).

Precision-Recall Tradeoff - Since only one of precision or recall can be improved, this metric offers a balanced measurement considering both precision and recall (Bajaj, 2023).

F1 score(s) - This metric is a combination of precision and recall (Bajaj, 2023). F1 scores can be calculated for each class (comparable to binary accuracy).

AUROC (Area under Receiver operating characteristics curve) - This metric measures true positive rates (TPR) and false positive rates (FPR) to determine the accuracy of the model (Draelos, 2019).



$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Figures 24-25: Descriptions of some performance metrics

In addition to these quantitative performance metrics, qualitative observation of the program is conducted periodically. Such observation detects possible malfunctions and analyzes the values in the confusion matrices. This goes hand-in-hand with most of the accuracy quantitative metrics, but provides a varied lens of design performance.

From the aforementioned quantitative and qualitative performance metrics or criteria, accuracy, average binary accuracy, average F1 score, and confusion matrices were utilized to assess the effectiveness of the machine learning model. These statistics holistically cover the likelihood of the model making a correct prediction and the likelihood of the model being correct when choosing a certain treatment.

Ultimately, if the results yield a high accuracy and a great balance between precision and recall, this machine learning simulation serves as a proof-of-concept for a model based on real patient data from hospitals. Such a model can eradicate or at least produce a great reduction in medication errors and pharmaceutical industry influence; by being institutionalized as a requirement for healthcare providers to complete in order to aid or guide their prescription decisions with unbiased input, this program has the potential to enhance the care of patients with Parkinson’s disease worldwide.

Design Plan and Methods

General Design Plan

- Importing Modules (Materials)
- Dataset
    - Generating Simulation Dataset (with Constraints)
    - Normalizing and Encoding Data
    - Splitting Dataset
- Model
    - Defining Plotting Function
    - Defining Create Model Function
    - Defining Train Model Function
    - Running Model

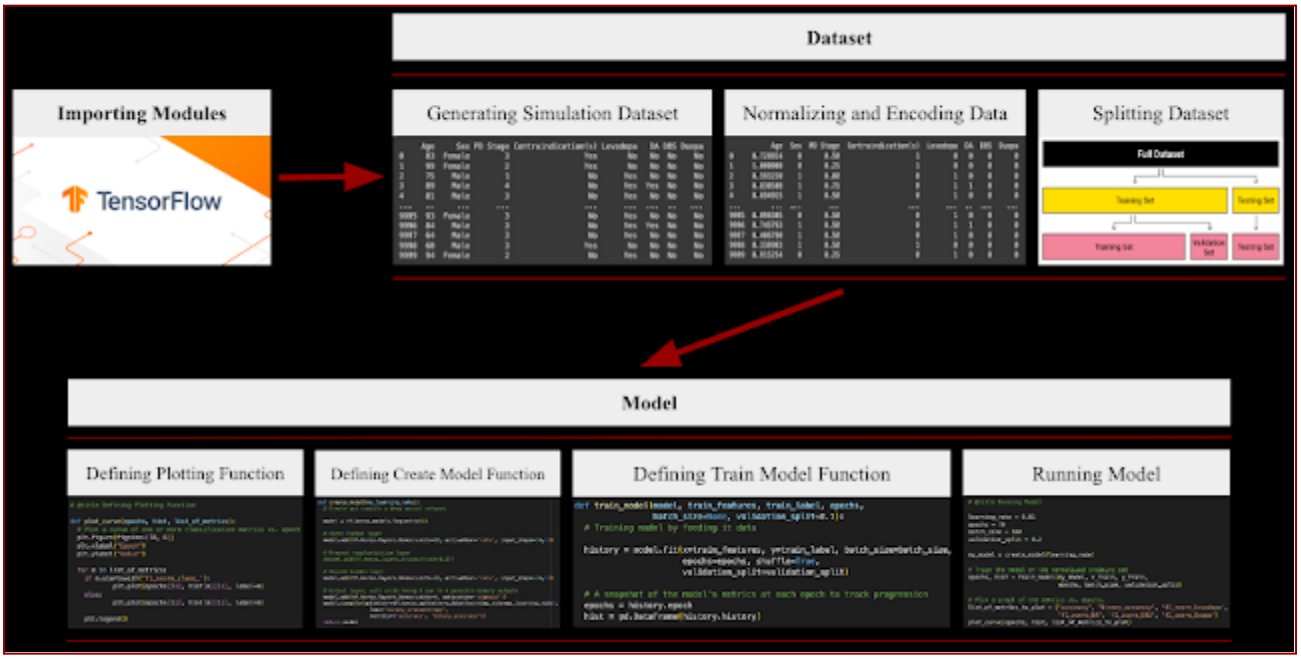


Figure 26: General design plan for ML model in this study

Importing Modules (Materials)

1. Create a Google Colab Jupyter Notebook to host the model
2. Import the necessary modules to the notebook, which include Python machine learning libraries, dataset analysis libraries, and data visualization libraries

```
# @title Importing Modules

import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, confusion_matrix
```

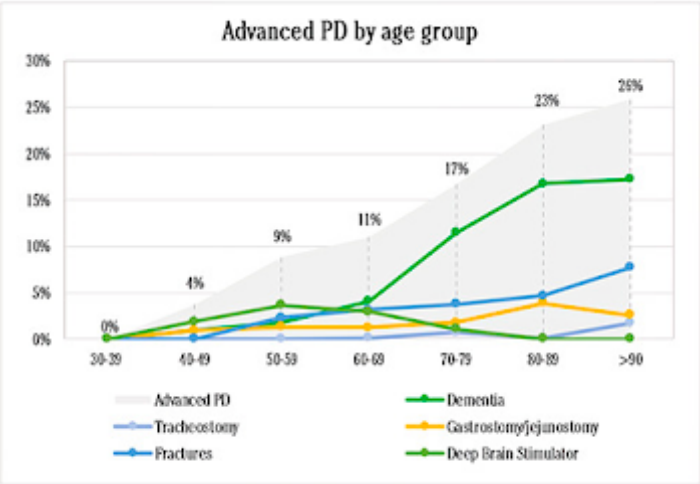
Generating Simulation Dataset

1. Research extensively statistics regarding the following factors of Parkinson's disease (PD) treatment: age, sex, PD stage, and comorbid or medication contraindications
    a. Age is the age of the patient
    b. Sex is the biological birth assignment of the patient
    c. PD stage is the Hoehn and Yahr Parkinson's disease scale, ranging from stage 1 to stage 5, inclusive (only integers will be considered)
    d. Contraindication factor will be presence or absence of comorbidities or related medications that may negatively interfere with the treatment prescribed for PD
        i. For PD, contraindicated comorbidities include psychiatric disorders and hypertension
        ii. Related contraindicated medications include antipsychotics (Chlorpromazine, Fluphenazine, Haloperidol, etc.), antihypertensives (Methyldopa), and antidepressants (Amoxapine) (American Parkinson Disease Association, n.d.)
2. Compile and consolidate data
    a. For this program, data from the following studies and articles were used for the creation of the dataset
        i. Treatment Patterns in Patients with Incident Parkinson's Disease in the United States (Houghton et al., 2019)
        ii. Incidence of Parkinson disease in North America (Willis et al., 2022)

iii. Parkinson's disease prevalence, age distribution and staging in Colombia (Orozco et al., 2020)
    1. Includes worldwide PD data
iv. Deep Brain Stimulation for Parkinson's Disease and Other Movement Disorders (Medical Advisory Secretariat, 2005)
v. Older age at DBS surgery raises risk of death over subsequent years (Lobo, 2023)
vi. Levodopa Carbidopa Intestinal Gel in Advanced Parkinson's Disease: DUOGLOBE [Duopa] Final 3-Year Results (Chaudhuri et al., 2023)

b. Shown below are examples of the data being used (calculations have to be made from such datasets to fit the model criteria)

Age (2015)		**Prevalence estimates per 100.000**	
	Global	**Male**	**Female**
30-39	2.9 (95% CI 1.8-4.45)	3.9 (95% CI 2.14-6.56)	1.9 (95% CI 0.73-3.96)
40-49	19.5 (95% CI 15.99-23.63)	25.7 (95% CI 19.98-32.51)	13.5 (95% CI 9.51-18.62)
50-59	91.6 (95% CI 82.71-101.29)	115.0 (95% CI 100.66-130.85)	70.1 (95% CI 59.46-82.16)
60-69	336.8 (95% CI 313.57-361.28)	408.8 (95% CI 371.70-448.61)	272.9 (95% CI 244.48-303.78)
70-79	1081.9 (95% CI 1019.27-1147.38)	1303.9 (95% CI 1200.95-1413.27)	906.5 (95% CI 830.36-987.84)
80-89	1786.1 (95% CI 1660.05-1919.19)	2236.9 (95% CI 2010.26-2482.16)	1508.6 (95% CI 1362.42-1666.26)
90+	1579.7 (95% CI 1305.37-1894.74)	1747.4 (95% CI 1269.68-2345.83)	1492.2 (95% CI 1167.58-1879.21)
All ages	157.92 (95% CI 152.25-163.43)	175.00 (95% CI 166.69-183.40)	141.91 (95% CI 134.85-149.24)



Sample characteristics	PD patients (n:3264)	Male (n:1750)	Female (n:1514)	p
Comorbidities (%)				
Median comorbidity count (IQR)	1 (1-2)	1 (1-2)	1 (0-2)	<0.001
Arterial hypertension	2142 (65.62)	1099 (62.80)	1043 (68.89)	<0.001
Diabetes	552 (16.91)	293 (16.74)	259 (17.11)	0.782
Dyslipidemia	314 (9.62)	154 (8.80)	160 (10.57)	0.088
COPD	342 (10.48)	172 (9.83)	170 (11.23)	0.193
Minimal cognitive impairment	50 (1.53)	30 (1.71)	20 (1.32)	0.362
Psychiatric disorders	717 (21.97)	333 (19.03)	384 (25.36)	<0.001
Sleep disorders	79 (2.42)	37 (2.11)	42 (2.77)	0.221
Stage (%)				
Advanced PD	512 (15.69)	232 (13.26)	280 (18.49)	<0.001
Dementia	313 (9.59)	144 (8.23)	169 (11.16)	0.005
Tracheostomy	12 (0.37)	8 (0.46)	4 (0.26)	0.364
Gastrostomy/jejunostomy	68 (2.08)	29 (1.66)	39 (2.58)	0.067
Fractures	120 (3.68)	45 (2.57)	75 (4.95)	<0.001
Deep brain stimulator	52 (1.59)	29 (1.66)	23 (1.52)	0.754

Hazard ratios for time between PD diagnosis and first treatment with:							
		Any treatment	**Levodopa/ combination**	**Dopamine agonists**	**MAO-Bi**	**Amantadine**	**Other dopaminergic agent**
Gender	Female	0.94 (0.92, 0.96)	0.92 (0.90, 0.94)	0.99 (0.96, 1.03)	0.82 (0.78, 0.85)	0.99 (0.94, 1.05)	0.79 (0.72, 0.86)
Age, years*	50–70	1.37 (1.30, 1.45)	1.56 (1.45, 1.66)	0.93 (0.86, 1.00)	0.89 (0.82, 0.97)	0.78 (0.70, 0.88)	0.84 (0.68, 1.03)
	>70	1.31 (1.24, 1.38)	2.06 (1.92, 2.20)	0.41 (0.38, 0.44)	0.33 (0.30, 0.36)	0.35 (0.31, 0.40)	0.66 (0.53, 0.81)
CCI §	1	0.95 (0.93, 0.98)	1.00 (0.97, 1.02)	0.84 (0.81, 0.88)	0.72 (0.68, 0.75)	0.85 (0.79, 0.91)	0.81 (0.72, 0.91)
	2	0.90 (0.88, 0.93)	0.95 (0.92, 0.98)	0.79 (0.76, 0.83)	0.65 (0.61, 0.69)	0.79 (0.72, 0.86)	0.71 (0.62, 0.82)
	3	0.81 (0.79, 0.83)	0.88 (0.86, 0.90)	0.68 (0.65, 0.71)	0.43 (0.40, 0.45)	0.68 (0.62, 0.73)	0.63 (0.56, 0.72)

First treatment class					Treated	Patients eligible for treatment pattern analysis (n=68,532)*	Patients treated (n=49,737)
Amantadine	Levodopa and levodopa combinations	Dopamine agonists	MAO-Bi	Other dopaminergic agents	n	%	%
	x				34,977	51.0	70.3
		x			7,729	11.3	15.5
			x		4,112	6.0	8.3
x					1,521	2.2	3.1
	x	x			521	0.8	1.0
	x		x		234	0.3	0.5
		x	x		148	0.2	0.3
x	x				132	0.2	0.3
	x			x	106	0.2	0.2
				x	100	0.1	0.2
Other combinations†					157	0.2	0.3

Figures 27-31: Examples of data utilized in simulated dataset for ML model

3. Organize statistics and associations in a spreadsheet
    a. In addition to statistics related to factors, the dataset will include “rules” to prevent misrepresentation and to reflect real world data
        i. Levodopa and Duopa treatments cannot both be assigned
        ii. If there is no potential presence of contraindication(s), at least one treatment should be assigned
        iii. If there is a potential presence of contraindication(s), no treatment should be assigned
    b. Shown below is the spreadsheet for this study

Features				Label	
Age	Sex	Stage of PD	Potential Contraindications	Treatment(s)*	Sources
40-49 (0.4%)	Male: 65.6% Female: 34.4%	Stages 1-3: 96% Stage 4: 4%	In Male: 13.03% In Female: 19.36%	If 1 ≤ PD Stage ≤ 2, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%; If 3 ≤ PD Stage ≤ 4, Levo: 63.4%, DA: 29.7%, DBS: 10.0%, Duopa: 5.0%	(Orozco et al., 2020) (Willis et al., 2022) (Conti et al., 2022) (Rizek et al., 2016) (Houghton et al., 2019) (Medical Advisory Secretariat, 2005)
50-59 (1.9%)	Male: 62.1% Female: 37.9%	Stages 1-3: 91% Stage 4: 9%	In Male: 16.03% In Female: 22.36%	If 1 ≤ PD Stage ≤ 2, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%; If 3 ≤ PD Stage ≤ 4, Levo: 69.6%, DA: 29.7%, DBS: 30.0%, Duopa: 7.5%	
60-69 (6.9%)	Male: 60.0% Female: 40.0%	Stages 1-3: 89% Stages 4-5: 11%	In Male: 19.03% In Female: 25.36%	If 1 ≤ PD Stage ≤ 2, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%; If 3 ≤ PD Stage ≤ 5, Levo: 75.8%, DA: 29.7%, DBS: 30.0%, Duopa: 10.0%	
70-79 (22.1%)	Male: 59.0% Female: 41.0%	Stages 1-3: 83% Stages 4-5: 17%	In Male: 19.03% In Female: 25.36%	If 1 ≤ PD Stage ≤ 2, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%; If 3 ≤ PD Stage ≤ 5, Levo: 82.0%, DA: 24.7%, DBS: 24.0%, Duopa: 10.0%	
80-89 (36.5%)	Male: 59.7% Female: 40.3%	Stages 2-3: 77% Stages 4-5: 23%	In Male: 22.03% In Female: 28.36%	If 2 ≤ PD Stage ≤ 3, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%; If 4 ≤ PD Stage ≤ 5, Levo: 82.0%, DA: 19.7%, DBS: 22.0%, Duopa: 7.5%	
90-99 (32.2%)	Male: 53.9% Female: 46.1%	Stages 2-3: 74% Stages 4-5: 26%	In Male: 25.03% In Female: 31.36%	If 2 ≤ PD Stage ≤ 3, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%; If 4 ≤ PD Stage ≤ 5, Levo: 81.0%, DA: 14.7%, DBS: 20.0%, Duopa: 5.0%	
Rules	1	2	3		
	Levodopa and Duopa cannot both be assigned	If there is no potential presence of contraindication(s), at least one treatment should be assigned	If there is a potential presence of contraindication(s), no treatment should be assigned	*Note: · Levo is Levodopa · LevoDA is combination of Levodopa and DA · Other combinations are present, but their distributions will not be implemented due to absence of such data	

Figure 32: Spreadsheet of statistical distributions for factors in simulated dataset

4. Consider constraints and assumptions in the simulated dataset (listed below are the constraints in this study)
    a. Due to the absence of a real dataset that contains precisely the factors needed, the dataset in this study had to accumulate statistics from studies in different periods of time and different locations
    b. Although the factors in this dataset (age, sex, PD stage, contraindications) are primarily what contribute to the optimal treatment plan for a patient, other minor factors such as a patient's metabolism of medications, availability of treatments, etc. were not considered
    c. Due to the fact that an extremely small percentage of PD patients are below 40 years old or above 99 years old, these ages were not considered in this dataset
    d. Advanced PD is considered stages 4 and 5 on the Hoehn and Yahr scale, while early PD is considered stages 1 and 2 on the Hoehn and Yahr scale
        i. Due to absence of statistics regarding middle PD (stage 3), middle PD was combined with early PD (stages 1-2)

ii. Due to unlikelihood of an individual aged 40-59 being in stage 5 of PD, advanced PD for ages 40-59 was only stage 4

iii. Due to unlikelihood of an individual aged 80-99 being in stage 1 of PD, early PD for ages 80-89 was only stages 2-3

e. Since most medications for psychiatric disorders are contraindicated while only one medication for hypertension is contraindicated (easily avoidable), the only contraindication considered was psychiatric disorders

f. Some of the statistical distributions over age had to be assumed, particularly potential contraindications and DBS/Duopa treatments

g. Statistics regarding treatments often involved treatments at onset, which in this dataset was assumed to be early or middle PD (stages 1-3)

h. Due to the fact that an extremely small percentage of patients with early PD have DBS or Duopa treatment, these treatments were not considered for simulated patients with early PD

i. Since no concrete statistics could be found associating sex with treatment for a PD patient, sex-related data was abandoned in the learning process of the model

j. Due to natural randomness associated with the generation of a simulated dataset, a minority of simulated patients may be unrealistic

5. Code the distributions and rules in the Google Colab Jupyter Notebook and create a Pandas DataFrame to store the data of 100000 simulated patients

```
# First Part

# @title Generating Simulation Dataset

# Datapoints
age = 0
sex = 0
pdStage = 0
contra = 0
levo = 0
levoCount = 0
dopAgo = 0
dopAgoCount = 0
dbs = 0
dbsCount = 0
duopa = 0
duopaCount = 0

# Seed the random functions for replicability
np.random.seed(100)

all_data = []

for x in range(100000):
```

```
# Second Part

# @title Generating Simulation Dataset

 if a >= 9.2 and a < 31.3:
  # Age: 70-79
  # Sex: 59.0% Male, 41.0% Female
  # PD Stage: 83% Stages 1-3, 17% Stages 4-5
  # Presence of Contraindications: 19.0% in Male, 25.4% in Female
  # Treatment(s):
  # If 1 ≤ PD Stage ≤ 2, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%;
  # If 3 ≤ PD Stage ≤ 5, Levo: 82.0%, DA: 24.7%, DBS: 24.0%, Duopa: 10.0%
  age = np.random.randint(low=70, high=80)
  if s < 59.0:
   sex = 1
  else:
   sex = 0
  if p < 83:
   pdStage = np.random.randint(low=1, high=4)
  else:
   pdStage = np.random.randint(low=4, high=6)
  if sex == 1:
   if c < 19.0:
```

```
# Variables
a = np.random.random() * 100
s = np.random.random() * 100
p = np.random.randint(100)
c = np.random.random() * 100
le = np.random.random() * 100
leDA = np.random.random() * 100
da = np.random.random() * 100
db = np.random.random() * 100
du = np.random.random() * 100

# Age: 40-49 is 0 <= a < 0.4, 50-59 is 0.4 <= a < 2.3,
# 60-69 is 2.3 <= a < 9.2, 70-79 is 9.2 <= a < 31.3,
# 80-89 is 31.3 <= a < 67.8, 90-100 is 67.8 <= a < 100
if a >= 0 and a < 0.4:
 # Age: 40-49
 # Sex: 65.6% Male, 34.4% Female
 # PD Stage: 96% Stages 1-3, 4% Stage 4
 # Presence of Contraindications: 13.0% in Male, 19.4% in Female
 # Treatment(s):
 # If 1 ≤ PD Stage ≤ 2, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%;
 # If 3 ≤ PD Stage ≤ 4, Levo: 63.4%, DA: 29.7%, DBS: 10.0%, Duopa: 5.0%
 age = np.random.randint(low=40, high=50)
 if s < 65.6:
  sex = 1
 else:
  sex = 0
 if p < 96:
  pdStage = np.random.randint(low=1, high=4)
 else:
  pdStage = 4
 if sex == 1:
  if c < 13.0:
   contra = 1
  else:
   contra = 0
 if sex == 0:
  if c < 19.4:
   contra = 1
  else:
   contra = 0
 if pdStage < 3:
  if le < 70.3:
   levo = 1
  if da < 15.5:
   dopAgo = 1
  if leDA < 14.2:
   levo = 1
   dopAgo = 1
 if pdStage >= 3:
  if le < 63.4:
   levo = 1
  if da < 29.7:
   dopAgo = 1
  if db < 10.0:
   dbs = 1
  if du < 5.0:
   duopa = 1
if a >= 0.4 and a < 2.3:
 # Age: 50-59
 # Sex: 62.1% Male, 37.9% Female
 # PD Stage: 91% Stages 1-3, 9% Stage 4
 # Presence of Contraindications: 16.0% in Male, 22.4% in Female
 # Treatment(s):
 # If 1 ≤ PD Stage ≤ 2, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%;
 # If 3 ≤ PD Stage ≤ 4, Levo: 69.6%, DA: 29.7%, DBS: 30.0%, Duopa: 7.5%
 age = np.random.randint(low=50, high=60)
 if s < 62.1:
  sex = 1
 else:
  sex = 0
 if p < 91:
  pdStage = np.random.randint(low=1, high=4)
 else:
  pdStage = 4
 if sex == 1:
  if c < 16.0:
   contra = 1
  else:
   contra = 0
 if sex == 0:
  if c < 25.4:
   contra = 1
  else:
   contra = 0
 if pdStage < 3:
  if le < 70.3:
   levo = 1
  if da < 15.5:
   dopAgo = 1
  if leDA < 14.2:
   levo = 1
   dopAgo = 1
 if pdStage >= 3:
  if le < 82.0:
   levo = 1
  if da < 24.7:
   dopAgo = 1
  if db < 24.0:
   dbs = 1
  if du < 10.0:
   duopa = 1
if a >= 31.3 and a < 67.8:
 # Age: 80-89
 # Sex: 59.7% Male, 40.3% Female
 # PD Stage: 77% Stages 2-3, 23% Stages 4-5
 # Presence of Contraindications: 22.0% in Male, 28.4% in Female
 # Treatment(s):
 # If 2 ≤ PD Stage ≤ 3, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%;
 # If 4 ≤ PD Stage ≤ 5, Levo: 82.0%, DA: 19.7%, DBS: 22.0%, Duopa: 7.5%
 age = np.random.randint(low=80, high=90)
 if s < 59.7:
  sex = 1
 else:
  sex = 0
 if p < 77:
  pdStage = np.random.randint(low=2, high=4)
 else:
  pdStage = np.random.randint(low=4, high=6)
 if sex == 1:
  if c < 22.0:
   contra = 1
  else:
   contra = 0
 if sex == 0:
  if c < 28.4:
   contra = 1
  else:
   contra = 0
 if pdStage < 4:
  if le < 70.3:
   levo = 1
  if da < 15.5:
   dopAgo = 1
  if leDA < 14.2:
   levo = 1
   dopAgo = 1
 if pdStage >= 4:
  if le < 82.0:
   levo = 1
  if da < 19.7:
   dopAgo = 1
  if db < 22.0:
   dbs = 1
  if du < 7.5:
   duopa = 1
if a >= 67.8 and a < 100:
 # Age: 90-99
```

```
    if p < 91:
     pdStage = np.random.randint(low=1, high=4)
    else:
     pdStage = 4
    if sex == 1:
     if c < 16.0:
      contra = 1
     else:
      contra = 0
    if sex == 0:
     if c < 22.4:
      contra = 1
     else:
      contra = 0
    if pdStage < 3:
     if le < 70.3:
      levo = 1
     if da < 15.5:
      dopAgo = 1
     if leDA < 14.2:
      levo = 1
      dopAgo = 1
    if pdStage >= 3:
     if le < 69.6:
      levo = 1
     if da < 29.7:
      dopAgo = 1
     if db < 30.0:
      dbs = 1
     if du < 7.5:
      duopa = 1
  if a >= 2.3 and a < 9.2:
   # Age: 60-69
   # Sex: 60.0% Male, 40.0% Female
   # PD Stage: 89% Stages 1-3, 11% Stages 4-5
   # Presence of Contraindications: 19.0% in Male, 25.4% in Female
   # Treatment(s):
   # If 1 ≤ PD Stage ≤ 2, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%;
   # If 3 ≤ PD Stage ≤ 5, Levo: 75.8%, DA: 29.7%, DBS: 30.0%, Duopa: 10.0%
   age = np.random.randint(low=60, high=70)
   if s < 60.0:
    sex = 1
   else:
    sex = 0
   if p < 89:
    pdStage = np.random.randint(low=1, high=4)
   else:
    pdStage = np.random.randint(low=4, high=6)
   if sex == 1:
    if c < 19.0:
     contra = 1
    else:
     contra = 0
   if sex == 0:
    if c < 25.4:
     contra = 1
    else:
     contra = 0
   if pdStage < 3:
    if le < 70.3:
     levo = 1
    if da < 15.5:
     dopAgo = 1
    if leDA < 14.2:
     levo = 1
     dopAgo = 1
   if pdStage >= 3:
    if le < 75.8:
     levo = 1
    if da < 29.7:
     dopAgo = 1
   # Sex: 53.9% Male, 46.1% Female
   # PD Stage: 74% Stages 2-3, 26% Stages 4-5
   # Presence of Contraindications: 25.0% in Male, 31.4% in Female
   # Treatment(s):
   # If 2 ≤ PD Stage ≤ 3, Levo: 70.3%, DA: 15.5%, LevoDA: 14.2%;
   # If 4 ≤ PD Stage ≤ 5, Levo: 81.0%, DA: 14.7%, DBS: 20.0%, Duopa: 5.0%
   age = np.random.randint(low=90, high=100)
   if s < 53.9:
    sex = 1
   else:
    sex = 0
   if p < 74:
    pdStage = np.random.randint(low=2, high=4)
   else:
    pdStage = np.random.randint(low=4, high=6)
   if sex == 1:
    if c < 25.0:
     contra = 1
    else:
     contra = 0
   if sex == 0:
    if c < 31.4:
     contra = 1
    else:
     contra = 0
   if pdStage < 4:
    if le < 70.3:
     levo = 1
    if da < 15.5:
     dopAgo = 1
    if leDA < 14.2:
     levo = 1
     dopAgo = 1
   if pdStage >= 4:
    if le < 81.0:
     levo = 1
    if da < 14.7:
     dopAgo = 1
    if db < 20.0:
     dbs = 1
    if du < 5.0:
     duopa = 1

# Rules
if levo == 0 and dopAgo == 0 and dbs == 0 and duopa == 0:
 levo = 1
 dopAgo = 1
if contra == 1:
 levo = 0
 dopAgo = 0
 dbs = 0
 duopa = 0
if levo == 1 and duopa == 1:
 levo = 0

# Data Table Value Formatting
if sex == 1:
 sex = 'Male'
if sex == 0:
 sex = 'Female'
if contra == 1:
 contra = 'Yes'
else:
 contra = 'No'
if levo == 1:
 levo = 'Yes'
else:
 levo = 'No'
if dopAgo == 1:
 dopAgo = 'Yes'
else:
```

```
if db < 30.0:
 dbs = 1
if du < 10.0:
 duopa = 1
```

```
 dopAgo = 'No'
if dbs == 1:
 dbs = 'Yes'
else:
 dbs = 'No'
if duopa == 1:
 duopa = 'Yes'
else:
 duopa = 'No'

 indpat_data = [age, sex, pdStage, contra, levo, dopAgo, dbs, duopa]
 all_data.append(indpat_data)

patient_data = np.array(all_data)

columns = ['Age', 'Sex', 'PD Stage', 'Contraindication(s)', 'Levodopa', 'DA', 'DBS',
'Duopa']

df = pd.DataFrame(data=patient_data, columns=columns)
```

6. Shown below is an example of a printed DataFrame after running the above code

	Age	Sex	PD Stage	Contraindication(s)	Levodopa	DA	DBS	Duopa
0	83	Female	3	Yes	No	No	No	No
1	99	Female	2	Yes	No	No	No	No
2	75	Male	1	No	Yes	No	No	No
3	89	Male	4	No	Yes	Yes	No	No
4	81	Male	3	No	Yes	No	No	No
...	..	...	...	...	...	...	..	...
9995	93	Female	3	No	Yes	No	No	No
9996	84	Male	3	No	Yes	Yes	No	No
9997	64	Male	3	No	Yes	No	No	No
9998	60	Male	3	Yes	No	No	No	No
9999	94	Female	2	No	Yes	No	No	No

Figure 33: Example of initial DataFrame of simulated data

Normalizing and Encoding Data

1. In order for the machine learning model to process each factor with an equal weight, scale the values of age and PD stage to be in the range of 0 to 1, inclusive
2. Encode sex, contraindication(s), and treatments (Levodopa, DA, DBS, Duopa) into binary values

```
# @title Normalizing and Encoding Data

normalized_df = df.copy();

normalized_df['Age'] = normalized_df['Age'].astype(float)
```

```
normalized_df['PD Stage'] = normalized_df['PD Stage'].astype(float)

# Normalizing Data
normalized_df['Age'] = (normalized_df['Age'] - 40) / 59
normalized_df['PD Stage'] = (normalized_df['PD Stage'] - 1) / 4

# Encode 'Sex' column to binary
normalized_df['Sex'] = np.where(normalized_df['Sex'] == 'Male', 1, 0)

# Encode other columns to binary
binary_columns = ['Contraindication(s)', 'Levodopa', 'DA', 'DBS', 'Duopa']
for col in binary_columns:
 normalized_df[col] = np.where(normalized_df[col] == 'Yes', 1, 0)

print(normalized_df)
```

3. Shown below is an example of a printed normalized DataFrame after running the above code

	Age	Sex	PD Stage	Contraindication(s)	Levodopa	DA	DBS	Duopa
0	0.728814	0	0.50	1	0	0	0	0
1	1.000000	0	0.25	1	0	0	0	0
2	0.593220	1	0.00	0	1	0	0	0
3	0.830508	1	0.75	0	1	1	0	0
4	0.694915	1	0.50	0	1	0	0	0
...	...	...	...	...	...	..	...	...
9995	0.898305	0	0.50	0	1	0	0	0
9996	0.745763	1	0.50	0	1	1	0	0
9997	0.406780	1	0.50	0	1	0	0	0
9998	0.338983	1	0.50	1	0	0	0	0
9999	0.915254	0	0.25	0	1	0	0	0

Figure 34: Example of normalized DataFrame of simulated data

Splitting Dataset

1. Utilize train_test_split function from the scikit machine learning library to shuffle and split the large dataset into the training set and the test set (training set makes up 80% of whole dataset, test set makes up 20% of whole dataset)

2. Set the first four columns (Age, Sex, PD stage, Contraindication(s)) to be the inputs/features (x) and the latter four columns (Levodopa, DA, DBS, and Duopa) to be the outputs/labels (y) for training and test sets

```
# @title Splitting Dataset

# Shuffle and split dataset into training and test sets
train, test = train_test_split(normalized_df, test_size=0.2)

# Divide columns into inputs and outputs
x_train = train.iloc[:, 0:4]
y_train = train.iloc[:, 4:8]
x_test = test.iloc[:, 0:4]
y_test = test.iloc[:, 4:8]
```

Defining Plotting Function

1. Reference the TensorFlow "Multi-class classification with MNIST" public model to define the Matplotlib function that will graph the performance metrics over epochs when running the program

```
# @title Defining Plotting Function

def plot_curve(epochs, hist, list_of_metrics):
 # Plot a curve of one or more classification metrics vs. epoch
 plt.figure(figsize=(10, 6))
 plt.xlabel("Epoch")
 plt.ylabel("Value")

 for m in list_of_metrics:
  if m.startswith('f1_score_class_'):
      plt.plot(epochs[1:], hist[m][1:], label=m)
  else:
      plt.plot(epochs[1:], hist[m][1:], label=m)

  plt.legend()
```

Defining Create Model Function

1. Reference the TensorFlow "Multi-class classification with MNIST" public model to define the create_model function using the API TensorFlow Keras
2. Modify aspects of the function to include more layers, a dropout regularization layer (to ensure that model does not overfit data in the training set), more performance metrics (binary accuracy), etc.
    a. May be changed during testing to optimize performance
    b. Change output layer activation function to sigmoid to make classification model multi-label instead of multi-class

```
# @title Defining Create Model Function

def create_model(my_learning_rate):
  # Create and compile an artificial neural network

  model = tf.keras.models.Sequential()

  # First hidden layer
  model.add(tf.keras.layers.Dense(units=32, activation='relu', input_shape=(4,)))

  # Dropout regularization layer
  model.add(tf.keras.layers.Dropout(rate=0.2))

  # Second hidden layer
  model.add(tf.keras.layers.Dense(units=32,  activation='relu', input_shape=(4,)))

  # Output layer, with units being 4 due to 4 possible binary outputs
  model.add(tf.keras.layers.Dense(units=4, activation='sigmoid'))
 model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=my_learning_rate),
                loss='binary_crossentropy',
                metrics=['accuracy', 'binary_accuracy'])
  return model
```

3. Shown below is the general architecture of the neural network from the create_model function in this study
    a. Input layer is not shown
    b. Nodes on hidden layers are not to scale (32 nodes/units for each hidden layer)
    c. Dropout regularization layer is not included
    d. "None" is not an output in reality, but is a possibility if No is chosen for all of the treatments (occurs when there is a contraindication)

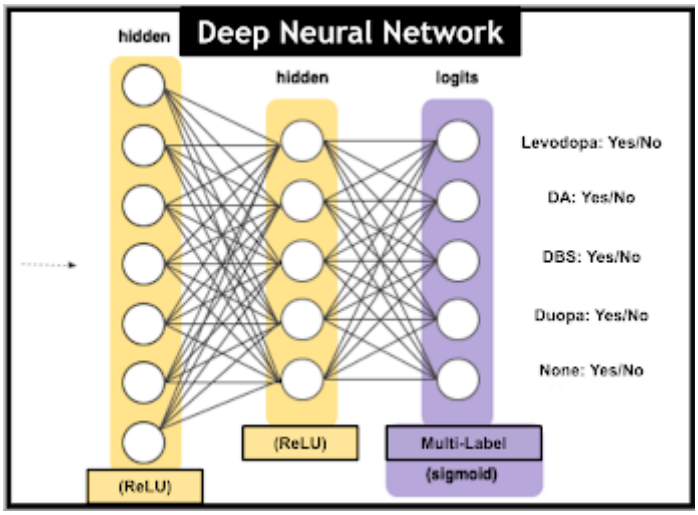


Figure 35: Example architecture of neural network in this study

Defining Train Model Function

1. Reference the TensorFlow "Multi-class classification with MNIST" public model to define the train_model function
2. Set the threshold for the sigmoid activation function in the output layer to be 0.5 for both the training and test sets
    a. If the output value for each class/label (Levodopa, DA, DBS, Duopa) exceeds 0.5, the model will predict Yes for that class
3. Use the scikit machine learning library f1_score function to evaluate the F1 scores for each class
    a. Calculate the average F1 score for all classes using the same function (macro-average)

4. Use the scikit machine learning library confusion_matrix function to calculate the confusion matrix for each class when evaluating the model on the test set
    a. Use the Seaborn data visualization library to plot and display these confusion matrices in the form of a heatmap

```
# @title Defining Train Model Function

def train_model(model, train_features, train_label, epochs,
       batch_size=None, validation_split=0.1):
 # Training model by feeding it data

 history = model.fit(x=train_features, y=train_label, batch_size=batch_size,
          epochs=epochs, shuffle=True,
          validation_split=validation_split)

 # A snapshot of the model's metrics at each epoch to track progression
 epochs = history.epoch
 hist = pd.DataFrame(history.history)

 # Setting model prediction threshold for sigmoid activation function in training set
 y_pred_prob = model.predict(train_features)
 y_pred = (y_pred_prob > 0.5).astype(int)

 f1_scores = f1_score(train_label, y_pred, average=None)

 # Calculate F1 score for each class/label
 for i, score in enumerate(f1_scores):
  hist[f'f1_score_{y_train.columns[i]}'] = score

 # Calculate average F1 score
 macro_avg_f1_score = f1_score(train_label, y_pred, average='macro')

 # Print average F1 score
 print(f"Macro-average F1 Score on Training and Validation Sets:
{macro_avg_f1_score}")

 # Setting model prediction threshold for sigmoid activation function in test set
```

```
 y_pred_test_prob = my_model.predict(x_test)
 y_pred_test = (y_pred_test_prob > 0.5).astype(int)

 # Calculate average F1 score for the test set
 macro_avg_f1_score_test = f1_score(y_test, y_pred_test, average='macro')

 # Print average F1 score for the test set
 print(f"Macro-average F1 Score on the Test Set: {macro_avg_f1_score_test}")

 # Calculate confusion matrix for each class/label on the test set
 for i, column in enumerate(y_test.columns):
  class_labels = y_test[column].values
  class_predictions = y_pred_test[:, i]
  cm = confusion_matrix(class_labels, class_predictions)

  # Reverse the confusion matrix values to match the conventional format
  cm = cm[::-1, ::-1]

  # Define the order of labels in reverse to match the conventional format
  labels_order = ['1', '0']

  # Plot the confusion matrix using Seaborn sns
  plt.figure(figsize=(6, 4))
  sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', linewidths=0.5, linecolor='black',
cbar=False,
        xticklabels=labels_order, yticklabels=labels_order)
  plt.title(f'Confusion Matrix for class {column} (Test Set)')
  plt.xlabel('Predicted Label')
  plt.ylabel('True Label')
  plt.show()

 return epochs, hist
```

Running Model

1. Establish the following hyperparameters: learning rate, epochs, batch size, and validation split

   a. Hyperparameters will be modified and tuned during testing for optimized performance
2. Use create and train model functions in accordance with the parameters
3. Plot the following performance metrics over epochs: accuracy, binary accuracy, and F1 score for each class
4. Evaluate the performance of the model on the test set

```
# @title Running Model

# Hyperparameters
learning_rate = 0.01
epochs = 100
batch_size = 500
validation_split = 0.2

my_model = create_model(learning_rate)

# Train the model on the normalized training set
epochs, hist = train_model(my_model, x_train, y_train,
             epochs, batch_size, validation_split)

# Plot a graph of the metrics vs. epochs.
list_of_metrics_to_plot = ['accuracy', 'binary_accuracy', 'f1_score_Levodopa',
             'f1_score_DA', 'f1_score_DBS', 'f1_score_Duopa']
plot_curve(epochs, hist, list_of_metrics_to_plot)

# Evaluate against the test set.
print("\n Evaluate the new model against the test set:")
my_model.evaluate(x=x_test, y=y_test, batch_size=batch_size)
```

Prototype 1 (Original Neural Network)

Description

The artificial neural network for Prototype 1 followed exactly the protocol of the design plan and methods in the previous section.

Note that the hyperparameters were tuned prior to recording data in testing. The only changes made between versions were related to the architecture of the ANN (number of hidden layers, number of nodes, presence or absence of dropout regularization layer).

Also note that there is nearly zero variability in results when running the same model version simply due to the machine learning process remaining unchanged. Thus, results from only one trial of each version were collected, invalidating statistical tests.

Results

Version 1 (1 HL)

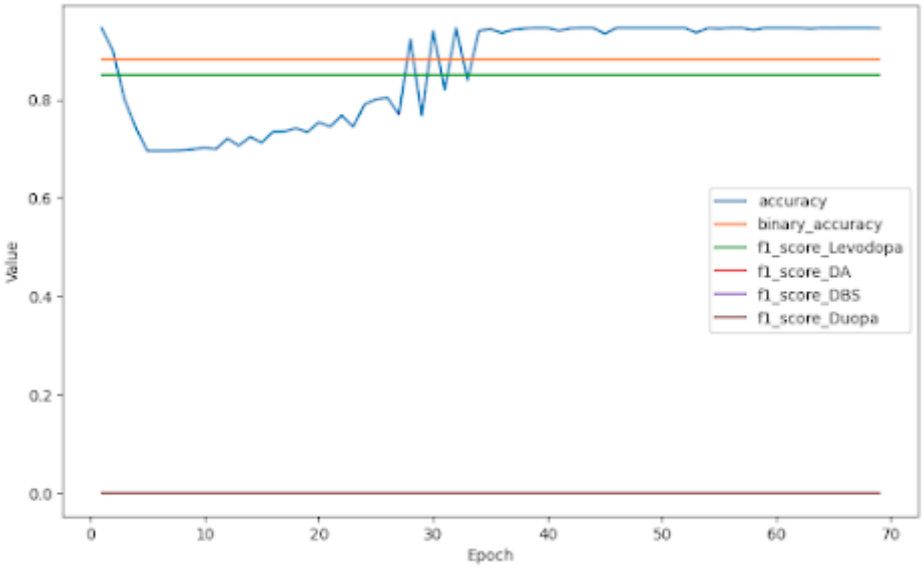
Model Specifics

- Architecture
    - Hidden Layers (HL): 1
        - First Layer Nodes: 32
        - Second Layer Nodes: N/A
    - Dropout Regularization Layer (DR): No
- Hyperparameters
    - Learning Rate: 0.01
    - Epochs: 70
    - Batch Size: 500
    - Validation Split: 0.2

Performance Criteria

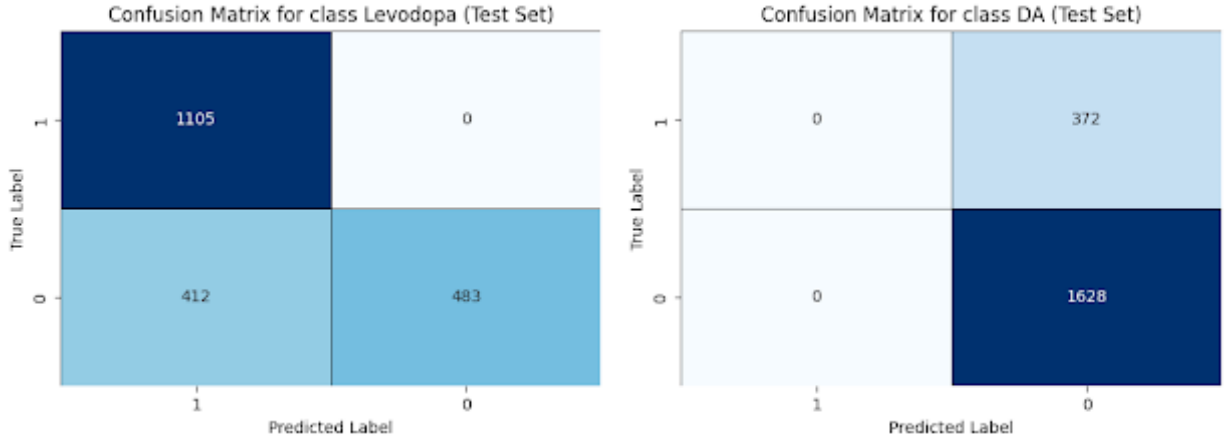
- Training and Validation Sets Evaluation
    - Macro-average F1 Score: 21.25%

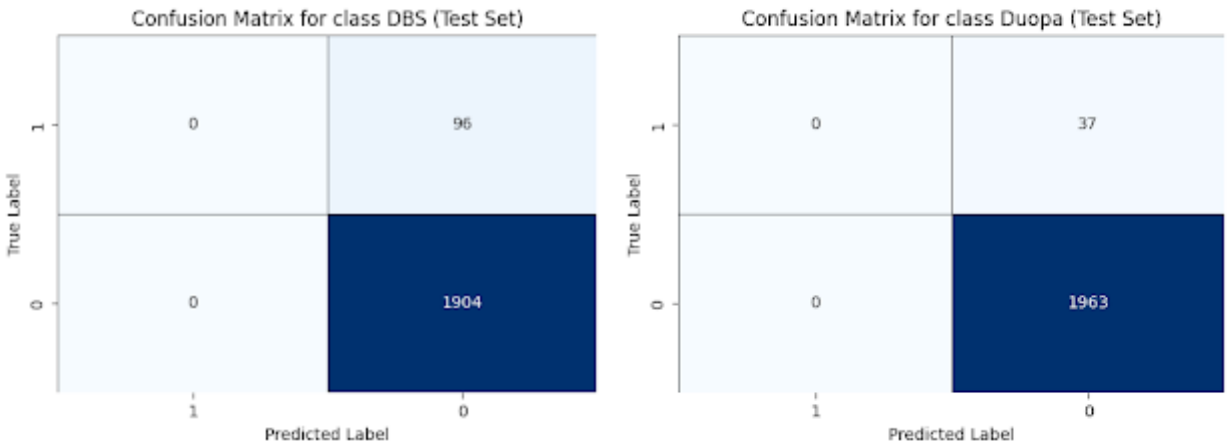
- Graph of Accuracy, Binary Accuracy, and Class F1 Scores over Epochs (shown below)



Graph 1.1: Accuracy, binary accuracy, and class F1 scores over epochs (Version 1)

- Test Set Evaluation
  - Loss: 26.18%
  - Accuracy: 94.40%
  - Binary Accuracy: 88.54%
  - Macro-average F1 Score: 21.07%
  - Confusion Matrices (shown below)





Confusion Matrices 1.1-1.4: Model predictions of each treatment class (Version 1)

- Observations
    - Accuracy reaching convergence
    - Binary accuracy constant
    - When model predicts Yes (1) in test set (per class)
        - Levodopa: Very Often (75.9%)
        - DA: Never (0%)
        - DBS: Never (0%)
        - Duopa: Never (0%)

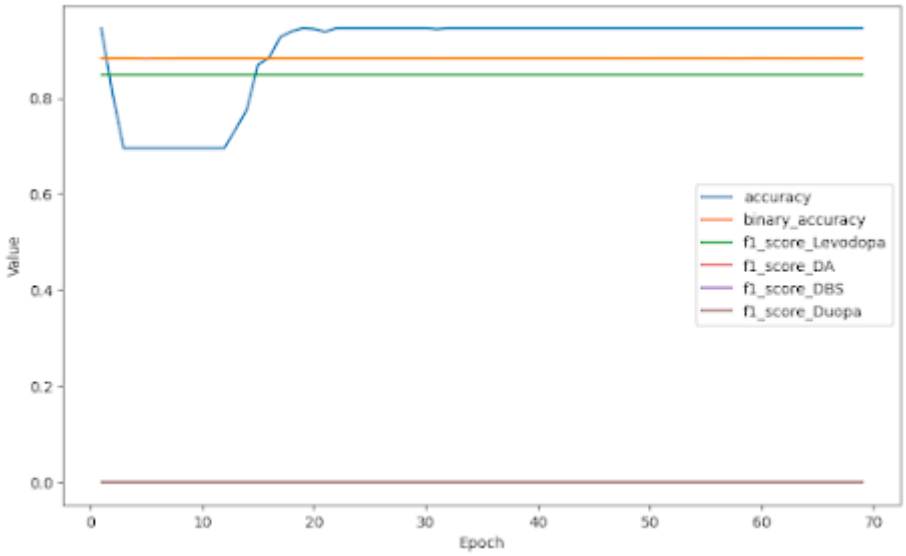
Version 2 (2 HL)

Model Specifics

- Architecture
    - Hidden Layers (HL): 2
        - First Layer Nodes: 32
        - Second Layer Nodes: 32
    - Dropout Regularization Layer (DR): No
- Hyperparameters
    - Learning Rate: 0.01
    - Epochs: 70
    - Batch Size: 500
    - Validation Split: 0.2

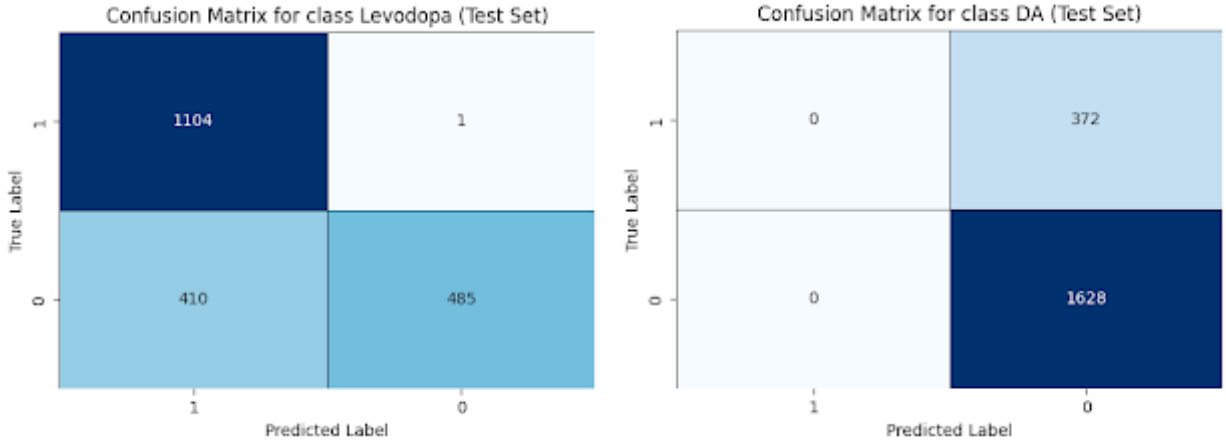
Performance Criteria

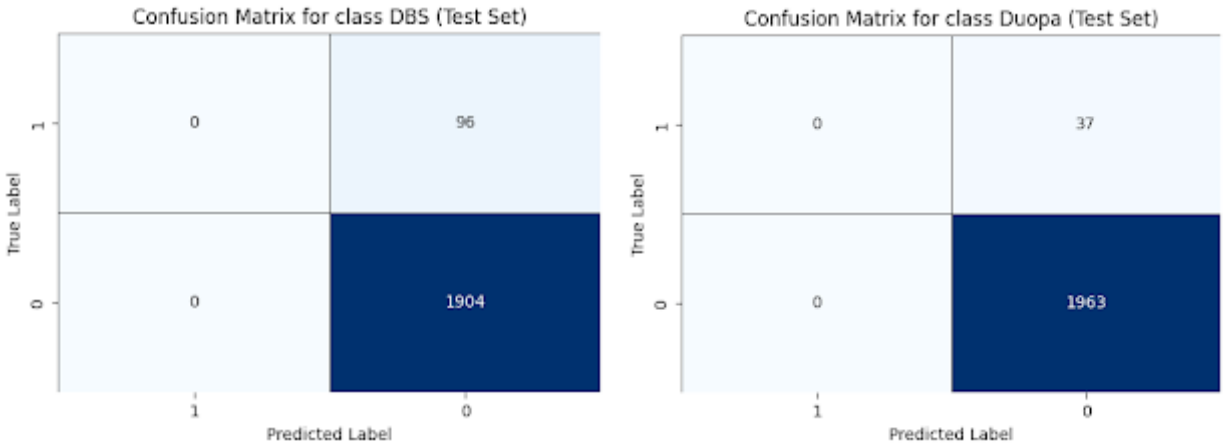
- Training and Validation Sets Evaluation
    - Macro-average F1 Score: 21.24%
    - Graph of Accuracy, Binary Accuracy, and Class F1 Scores over Epochs (shown below)



Graph 1.2: Accuracy, binary accuracy, and class F1 scores over epochs (Version 2)

- Test Set Evaluation
    - Loss: 26.16%
    - Accuracy: 94.40%
    - Binary Accuracy: 88.55%
    - Macro-average F1 Score: 21.08%
    - Confusion Matrices (shown below)





Confusion Matrices 1.5-1.8: Model predictions of each treatment class (Version 2)

- Observations
    - Accuracy reaching convergence
    - Binary accuracy constant
    - When model predicts Yes (1) in test set
        - Levodopa: Very Often (75.9%)
        - DA: Never (0%)
        - DBS: Never (0%)
        - Duopa: Never (0%)

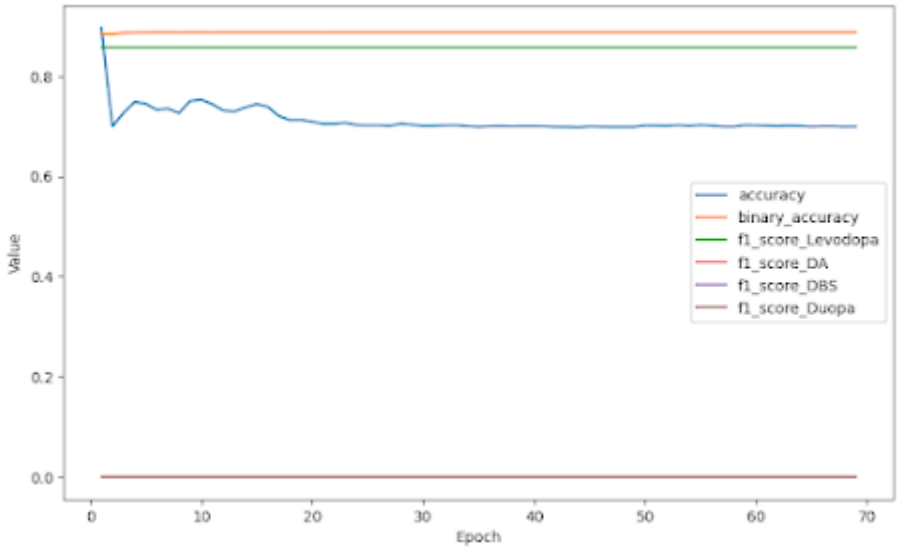
Version 3 (2 HL, DR)

Model Specifics

- Architecture
    - Hidden Layers (HL): 2
        - First Layer Nodes: 32
        - Second Layer Nodes: 32
    - Dropout Regularization Layer (DR): Yes
- Hyperparameters
    - Learning Rate: 0.01
    - Epochs: 70
    - Batch Size: 500
    - Validation Split: 0.2

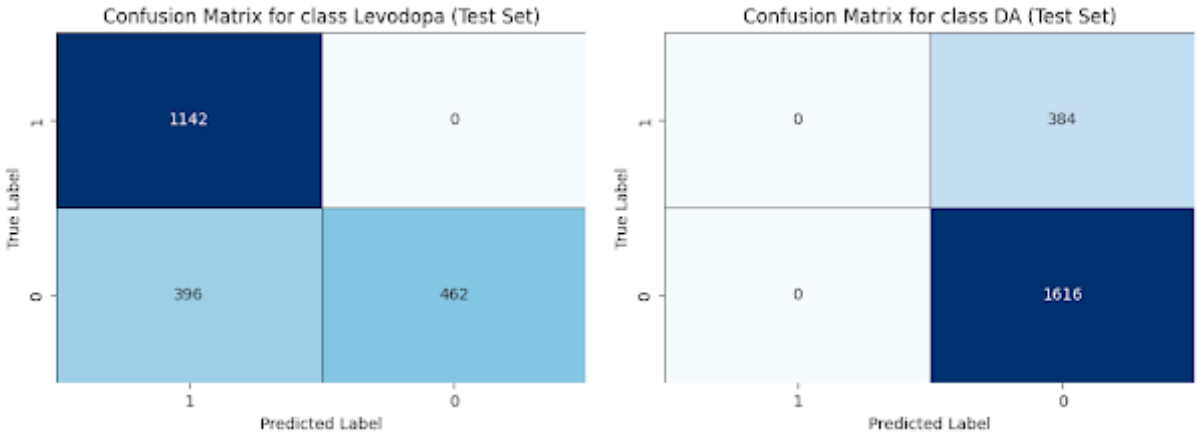
Performance Criteria

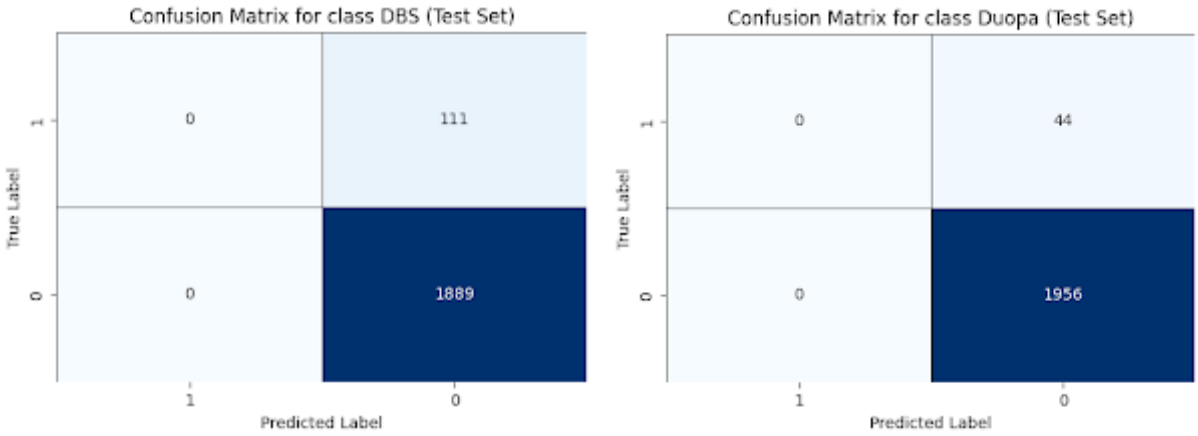
- Training and Validation Sets Evaluation
    - Macro-average F1 Score: 21.45%
    - Graph of Accuracy, Binary Accuracy, and Class F1 Scores over Epochs (shown below)



Graph 1.3: Accuracy, binary accuracy, and class F1 scores over epochs (Version 3)

- Test Set Evaluation
    - Loss: 26.91%
    - Accuracy: 71.20%
    - Binary Accuracy: 88.31%
    - Macro-average F1 Score: 21.31%
    - Confusion Matrices (shown below)





Confusion Matrices 1.9-1.12: Model predictions of each treatment class (Version 3)

- Observations
  - Accuracy reaching convergence
  - Binary accuracy constant
  - When model predicts Yes (1) in test set
    - Levodopa: Very Often (76.9%)
    - DA: Never (0%)
    - DBS: Never (0%)
    - Duopa: Never (0%)

Table 1.1: Summary of Prototype 1 Results (Test Set Evaluation)

Prototype 1 Results*	Accuracy (%)	Binary Accuracy (%)	Average F1 Score (%)
Version 1 (1 HL)	94.40	88.54	21.07
Version 2 (2 HL)	94.40	88.55	21.08
Version 3 (2 HL, DR)	71.20	88.31	21.31

Discussion

Based on these results, the most effective model was that of Version 2 (two hidden layers), with the highest average of accuracy and binary accuracy (91.47%) as well as the second highest macro-average F1 score (21.08%) during the test set evaluation. However, the difference

in these performance metrics was insignificant when compared to those in Version 1 (one hidden layer). Specifically, the difference between Versions 1 and 2 was 0% in accuracy and 0.01% in binary accuracy and macro-average F1 score. This suggests that adding hidden layers to this ANN does not improve its performance. Version 3 (two hidden layers, dropout regularization layer) had a significant drop-off in terms of accuracy, but remained similar in binary accuracy and F1 scores. The dropout regularization is likely allowing less learning due to the model being exposed to less data each epoch.

The model appears to perform well at first, with both accuracy metrics above 88% in Version 2. However, upon further examination, the F1 scores, macro-average F1 score, and confusion matrix values indicate otherwise. The class F1 scores and the macro-average F1 score are quite low, suggesting low precision and recall. In fact, the only class F1 score above 0% is Levodopa. In other words, the model is not predicting true positives at all for most classes. This is reassured by analyzing the confusion matrices, where it is shown that the model is only predicting Yes (positive) for the Levodopa class in the test set evaluation. For all the other classes, the model is only predicting No. Such a result is due to the realistic class imbalances in the data, with the number of Levodopa prescriptions vastly outweighing the number of all other prescriptions combined. The model is using this uneven distribution to optimize accuracy.

In all three versions, the model seems to start at a high accuracy and decrease or fluctuate over epochs. This may be due to the natural randomness present in the simulation dataset. By merely predicting Yes to Levodopa, the accuracy is directly proportional to the batch of Levodopa prescriptions in each epoch, causing fluctuations as the number of Levodopa prescriptions in the batches changes. This is a limitation of the simulation dataset and prevents some associations from being formed in the machine learning process.

In order to improve the model in F1 score while maintaining the simulation dataset and a high accuracy, fundamental changes must be made that significantly reduce class imbalance.

Prototype 2 (Neural Network with Class Weight Adjustments)

Description

One method to enhance class balance is adjusting class weights. Such adjustments penalize errors on the minority classes more heavily, encouraging the model to make better predictions for them. This model, Prototype 2, utilized the ratio of levodopa prescriptions to a certain class prescriptions for the weights. This was accomplished by incorporating count variables in the dataset for the number of times each treatment is prescribed. Then, a weight parameter was added to the train model function. The changes to the original code are highlighted and bolded below.

```
# @title Generating Simulation Dataset

# Code that creates simulated patient dataset...

# Counter
  if levo == 1:
    levoCount += 1
  if dopAgo == 1:
    dopAgoCount += 1
  if dbs == 1:
    dbsCount += 1
  if duopa == 1:
    duopaCount += 1

# DataFrame creation, normalization, and splitting...

# Defining plotting function and create model function...

# @title Defining Train Model Function

# Setting class weights
class_weights = {0: 1.0, 1: ((levoCount*1.0)/dopAgoCount), 2:
((levoCount*1.0)/dbsCount), 3: ((levoCount*1.0)/duopaCount)}

def train_model(model, train_features, train_label, epochs,
                batch_size=None, validation_split=0.1):

  history = model.fit(x=train_features, y=train_label, batch_size=batch_size,
                      epochs=epochs, shuffle=True,
```

```
                    validation_split=validation_split,
class_weight=class_weights)

# Rest of model...
```

Note that the hyperparameters were tuned prior to recording data in testing. The only changes made between versions were related to the architecture of the ANN (number of hidden layers, number of nodes, presence or absence of dropout regularization layer).

Also note that there is nearly zero variability in results when running the same model version simply due to the machine learning process remaining unchanged. Thus, results from only one trial of each version were collected, invalidating statistical tests.

Results

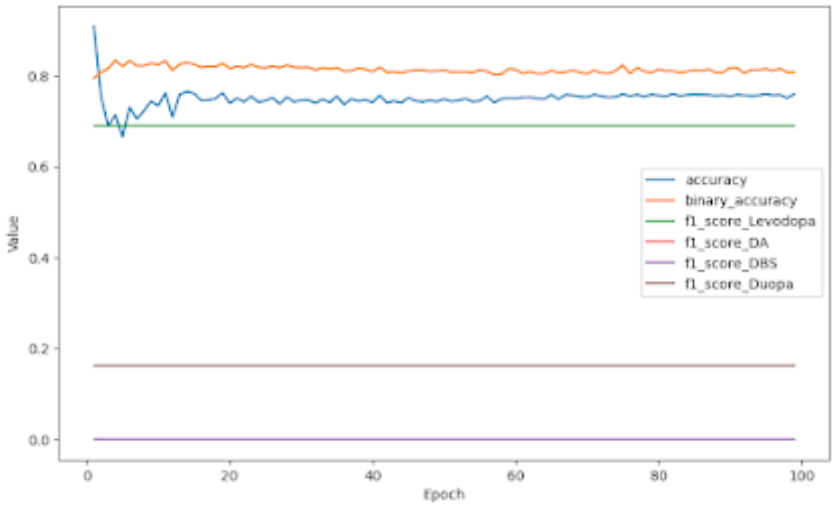
Version 1 (1 HL)

Model Specifics

- Architecture
    - Hidden Layers (HL): 1
        - First Layer Nodes: 32
        - Second Layer Nodes: N/A
    - Dropout Regularization Layer (DR): No
- Hyperparameters
    - Learning Rate: 0.01
    - Epochs: 100
    - Batch Size: 500
    - Validation Split: 0.2

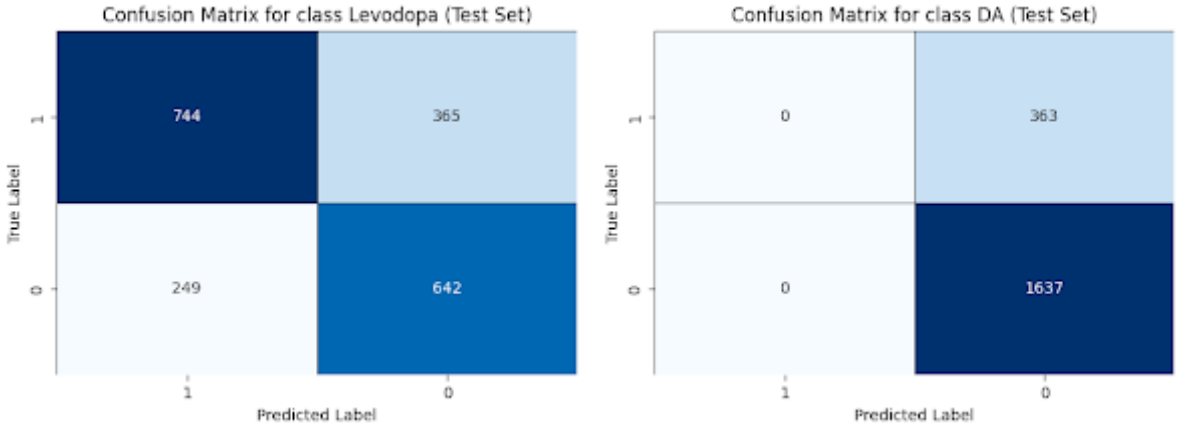
Performance Criteria

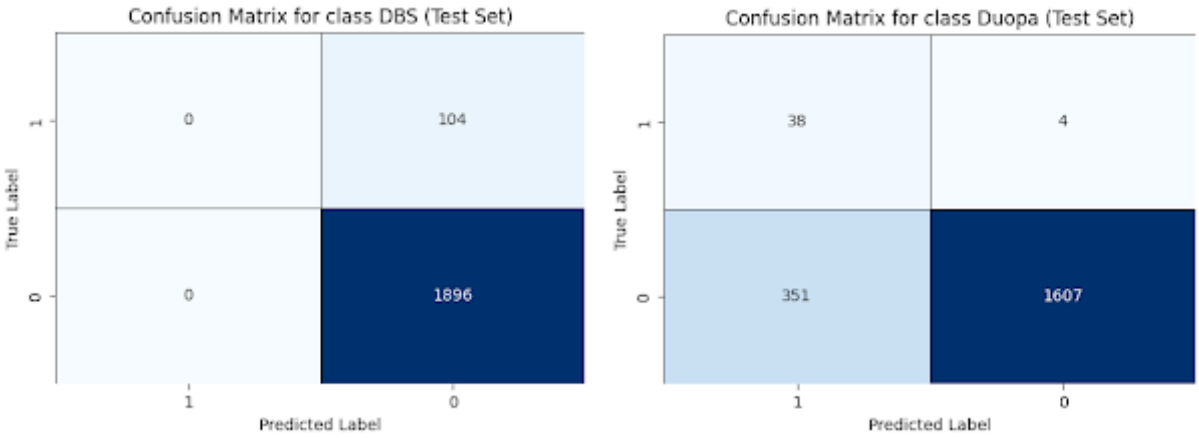
- Training and Validation Sets Evaluation
    - Macro-average F1 Score: 21.33%
    - Graph of Accuracy, Binary Accuracy, and Class F1 Scores over Epochs (shown below)



Graph 2.1: Accuracy, binary accuracy, and class F1 scores over epochs (Version 1)

- Test Set Evaluation
    - Loss: 31.50%
    - Accuracy: 77.70%
    - Binary Accuracy: 82.05%
    - Macro-average F1 Score: 22.11%
    - Confusion Matrices (shown below)





Confusion Matrices 2.1-2.4: Model predictions of each treatment class (Version 1)

- Observations
    - Accuracy fluctuating over epochs, but remaining relatively constant
    - Binary accuracy fluctuating, but remaining relatively constant
    - When model predicts Yes (1) in test set
        - Levodopa: Often (49.7%)
        - DA: Never (0%)
        - DBS: Never (0%)
        - Duopa: Infrequently (19.5%)

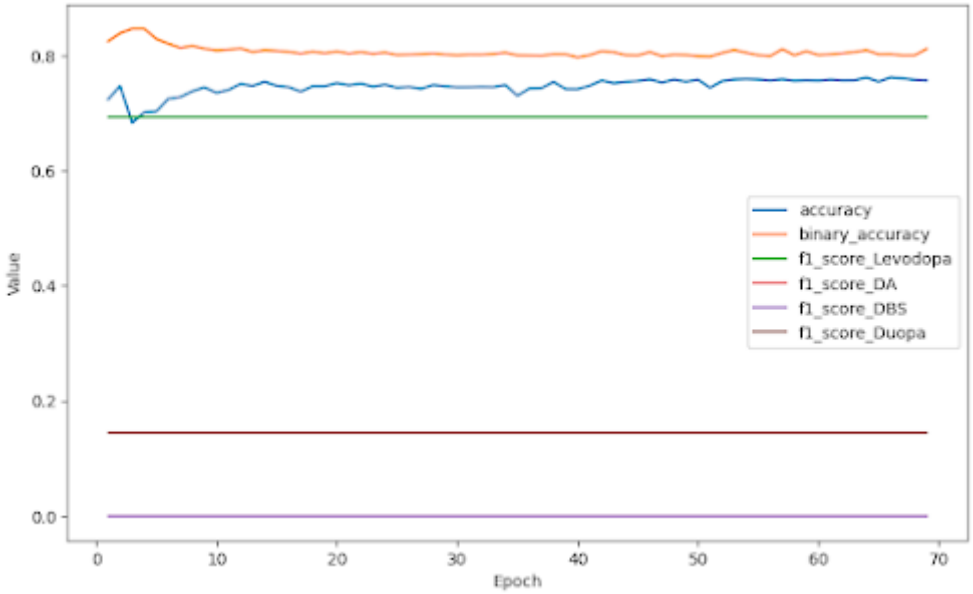
Version 2 (2 HL)

<u>Model Specifics</u>

- Architecture
    - Hidden Layers (HL): 2
        - First Layer Nodes: 32
        - Second Layer Nodes: 32
    - Dropout Regularization Layer (DR): No
- Hyperparameters
    - Learning Rate: 0.03
    - Epochs: 70
    - Batch Size: 250
    - Validation Split: 0.2

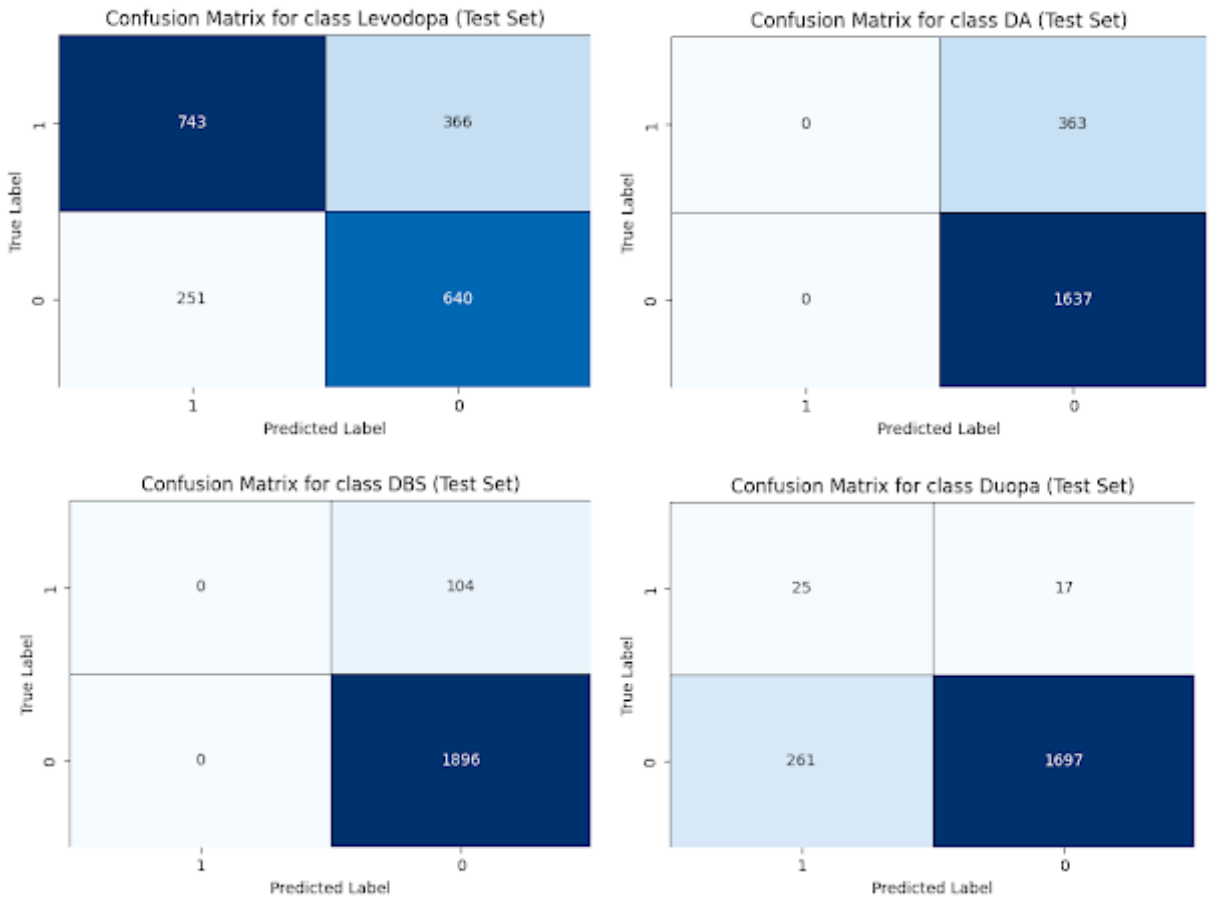
Performance Criteria

- Training and Validation Sets Evaluation
    - Macro-average F1 Score: 20.98%
    - Graph of Accuracy, Binary Accuracy, and Class F1 Scores over Epochs (shown below)



Graph 2.2: Accuracy, binary accuracy, and class F1 scores over epochs (Version 2)

- Test Set Evaluation
    - Loss: 30.74%
    - Accuracy: 79.05%
    - Binary Accuracy: 82.98%
    - Macro-average F1 Score: 21.48%
    - Confusion Matrices (shown below)



Confusion Matrices 2.5-2.8: Model predictions of each treatment class (Version 2)

- Observations
    - Accuracy remaining relatively constant over epochs
    - Binary accuracy remaining relatively constant
    - When model predicts Yes (1) in test set
        - Levodopa: Often (49.7%)
        - DA: Never (0%)
        - DBS: Never (0%)
        - Duopa: Infrequently (14.3%)

Version 3 (2 HL, DR)

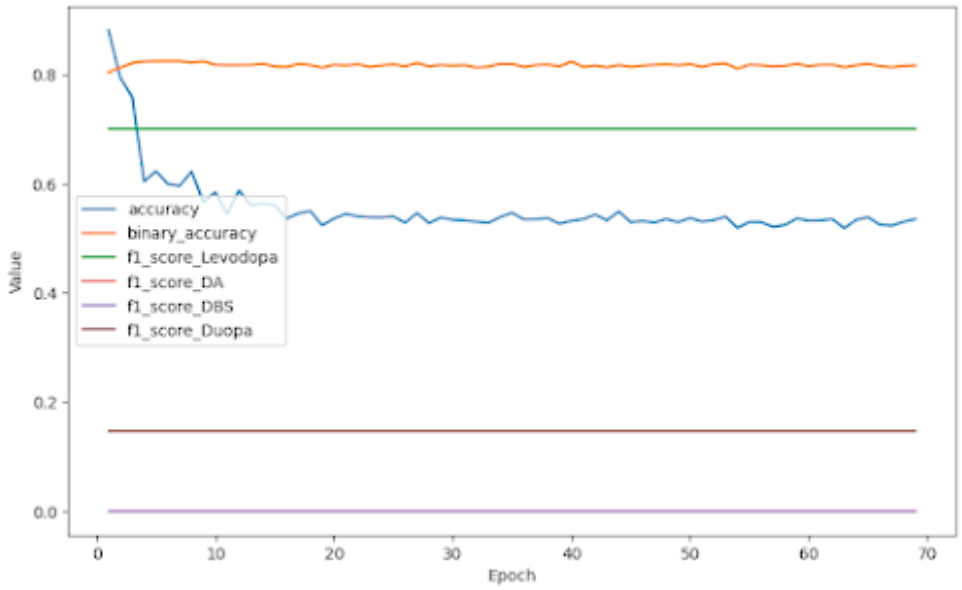
<u>Model Specifics</u>

- Architecture
    - Hidden Layers (HL): 2

        - First Layer Nodes: 32
        - Second Layer Nodes: 32
    - Dropout Regularization Layer (DR): Yes
- Hyperparameters
    - Learning Rate: 0.01
    - Epochs: 70
    - Batch Size: 500
    - Validation Split: 0.2

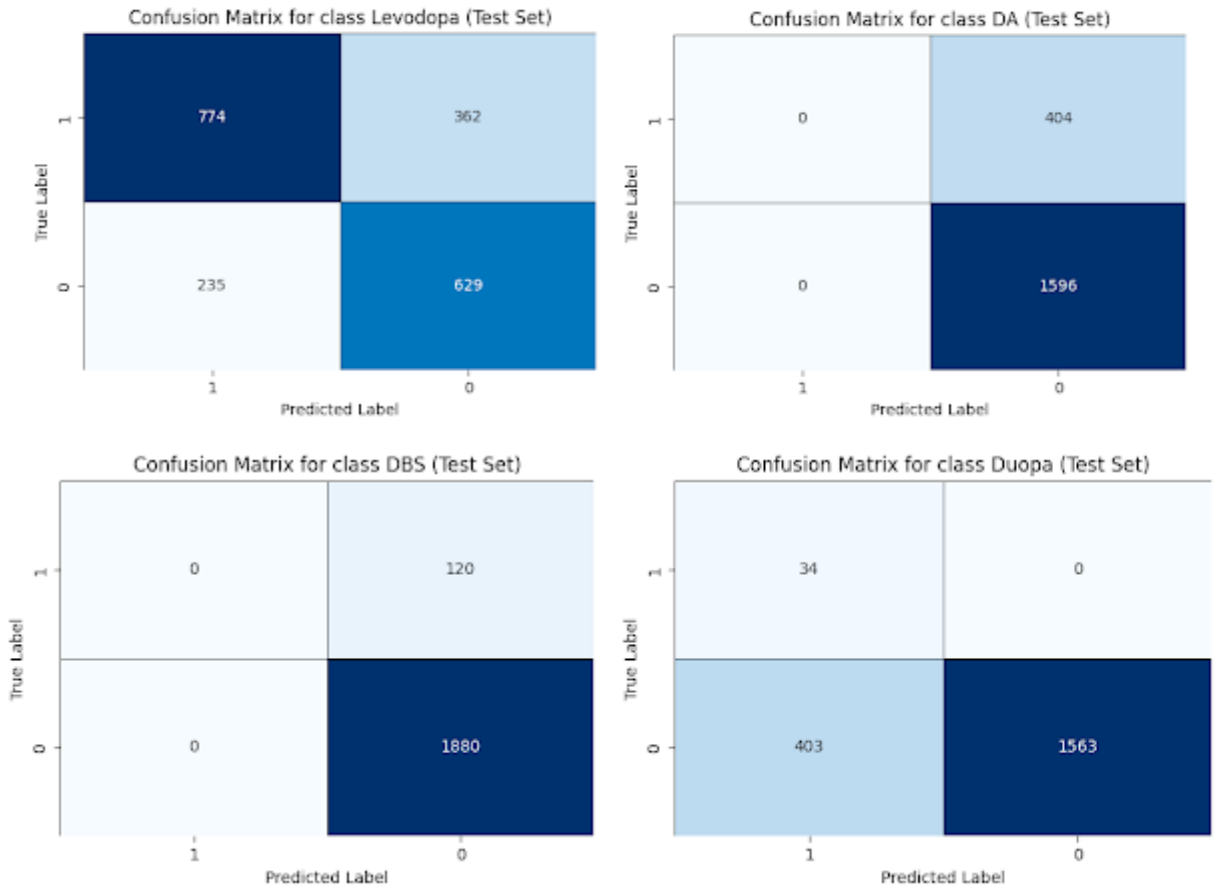
Performance Criteria

- Training and Validation Sets Evaluation
    - Macro-average F1 Score: 21.17%
    - Graph of Accuracy, Binary Accuracy, and Class F1 Scores over Epochs (shown below)



Graph 2.3: Accuracy, binary accuracy, and class F1 scores over epochs (Version 3)

- Test Set Evaluation
    - Loss: 34.36%
    - Accuracy: 51.75%
    - Binary Accuracy: 80.95%
    - Macro-average F1 Score: 21.65%
    - Confusion Matrices (shown below)



Confusion Matrices 2.9-2.12: Model predictions of each treatment class (Version 3)

- Observations
    - Accuracy fluctuating over epochs
    - Binary accuracy remaining relatively constant
    - When model predicts Yes (1) in test set
        - Levodopa: Very Often (55.95%)
        - DA: Never (0%)
        - DBS: Never (0%)
        - Duopa: Infrequently (21.85%)

Table 2.1: Summary of Prototype 2 Results (Test Set Evaluation)

Prototype 2 Results	Accuracy (%)	Binary Accuracy (%)	Average F1 Score (%)
Version 1 (1 HL)	77.70	82.05	22.11

Version 2 (2 HL)	79.05	82.98	21.48
Version 3 (2 HL, DR)	51.75	80.95	21.65

Table 2.2: Prototype Comparison of Optimal Versions

Prototype Comparison	Accuracy (%)	Binary Accuracy (%)	Average F1 Score (%)
Prototype 1 (Version 1)	94.40	88.54	21.07
Prototype 2 (Version 2)	79.05	82.98	21.48

Discussion

Based on these results, the most effective model was that of Version 2 (two hidden layers), with the highest average of accuracy and binary accuracy (81.02%). Version 1 (one hidden layer) and Version 3 (two hidden layers, dropout regularization layer) did have a higher macro-average F1 score (22.11% and 21.65%, respectively), but since accuracy is the more prominent measure of model efficacy, Version 2 was deemed more optimal. Nevertheless, the difference between one and two hidden layers from Versions 1 and 2 seems to be insignificant when comparing accuracy, binary accuracy, and macro-average F1 score during the test set evaluation. Version 3 (two hidden layers, dropout regularization layer) had a significant drop-off again in terms of accuracy, but remained similar in binary accuracy and F1 scores. The dropout regularization is likely allowing less learning due to the model being exposed to less data each epoch.

Unfortunately, the prototype comparison table demonstrates that the class weight adjustments in Prototype 2 only slightly improved the F1 score (0.41% increase) and caused a significant decrease in both accuracy metrics (15.35% decrease in accuracy, 5.56% decrease in binary accuracy) compared to Prototype 1. Moreover, as the confusion matrices indicate, while

the model in Prototype 2 can predict Yes for both Levodopa and Duopa, it continues to predict No every time for the other two classes, DA and DBS.

Further efforts to fix class imbalance or avert the randomness of the simulated dataset in the predictions must be made. Such changes can include data augmentation and sigmoid threshold adjustments. Data augmentation is a technique that transforms some of the data or adds data to represent minority classes more equally. In this project, data augmentation can be used by adding realistic cases of patients being prescribed treatments other than Levodopa. Sigmoid threshold adjustments can also represent the data more effectively by choosing the threshold that produces the best F1 score for each class (and thus best macro-average F1 score). Since higher F1 score correlates to higher precision and/or recall, adding this feature can be a step in the right direction.

Prototype 3 (Neural Network with Rules for Labels)

Description

The previous prototypes resulted in low class and average F1 scores, likely due to class imbalance as well as natural randomness from predictions based on statistics of the simulated dataset. One solution is a partially rules-based machine learning model that, in application, can be developed over time. This model format was implemented in Prototype 3, in which the features consisted of all the variables in the simulated dataset and the labels were determined primarily utilizing an example set of rules. Therefore, the features were the following: age, sex, stage of PD, presence of contraindications, taking L-Dopa (Levodopa), taking dopamine agonists, had DBS implant, had Duopa surgery. The labels were the optimal treatment(s) considering these aforementioned factors or features of the patient's medical history: L-Dopa (Levodopa), dopamine agonists, DBS, Duopa. Furthermore, the sample size of the dataset was increased to 100,000 simulated patients to increase model performance. Enacting this new structure of the DataFrame and input and output layers required changes to the code from Prototype 2. Some of these changes are highlighted and bolded below.

```
# @title Generating Simulation Dataset

# Features
age = 0
sex = 0
pdStage = 0
contra = 0
levoF = 0
dopAgoF = 0
dbsF = 0
duopaF = 0

# Labels (with counters)
levoL = 0
levoCount = 0
dopAgoL = 0
dopAgoCount = 0
```

```
dbsL = 0
dbsCount = 0
duopaL = 0
duopaCount = 0

# Beginning of simulated patient dataset creation...

for x in range(100000): # Increased sample size

  # Code that assigns values to feature variables...

  # Example Set of Rules (part of for-loop for dataset)
  if age <= 70 and pdStage < 3 and levoF == 1:
    levoL = 1
    dopAgoL = 1
  elif age <= 70 and pdStage < 3:
    dopAgoL = 1
  if age <= 70 and pdStage == 3 and levoF == 1:
    levoL = 1
  elif age <= 70 and pdStage == 3:
    dbsL = 1
  if age > 70 and age <= 80 and pdStage <= 3 and levoF == 1 and dopAgoF == 1:
    levoL = 1
    dopAgo = 1
  elif age > 70 and age <= 80 and pdStage <= 3 and levoF == 1:
    levoL = 1
  elif age > 70 and age <= 80 and pdStage <= 3 and dopAgoF == 1:
    dopAgoL = 1
  if age > 80 and pdStage <= 3 and levoF == 1:
    levoL = 1
  elif age > 80 and pdStage <= 3:
    duopaL = 1
  if age > 70 and pdStage > 3 and levoF == 1:
    levoL = 1
    dbsL = 1
  elif age > 70 and age <= 80 and pdStage > 3:
    duopaL = 1
    dbsL = 1


  # Adjustments (part of for-loop for dataset)
  if dbsF == 1:
    dbsL = 0
  if pdStage <= 3:
    duopaL = 0
    dbsL = 0
  if levoL == 1 and duopaL == 1:
```

```
    levoL = 0
  if levoL == 0 and dopAgoL == 0 and dbsL == 0 and duopaL == 0:
    levoL = 1
    dopAgoL = 1
  if contra == 1:
    levoL = 0
    dopAgoL = 0
    dbsL = 0
    duopaL = 0

# End of simulated dataset creation

# Adjustments to DataFrame creation, normalization, and splitting

# Adjustments to size of input layer in Create Model Function

# Rest of model...
```

	Age (F)	Sex (F)	PD Stage (F)	Contraindication(s) (F)	L-Dopa (F)	DA (F)	DBS (F)	Duopa (F)	L-Dopa (L)	DA (L)	DBS (L)	Duopa (L)
0	80	Male	4	No	Yes	Yes	Yes	No	Yes	No	No	No
1	65	Female	1	Yes	Yes	No	No	No	No	No	No	No
2	83	Female	2	No	Yes	No	No	No	Yes	No	No	No
3	81	Male	2	No	Yes	No	No	No	Yes	No	No	No
4	93	Female	2	Yes	Yes	Yes	No	No	No	No	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...
99995	74	Male	3	No	Yes	No	No	No	Yes	No	No	No
99996	91	Male	4	No	Yes	No	No	No	Yes	No	Yes	No
99997	58	Female	2	No	Yes	No	No	No	Yes	Yes	No	No
99998	84	Male	4	Yes	Yes	Yes	No	No	No	No	No	No
99999	77	Female	2	No	Yes	No	No	No	Yes	No	No	No

	Age (F)	Sex (F)	PD Stage (F)	Contraindication(s) (F)	L-Dopa (F)	DA (F)	DBS (F)	Duopa (F)	L-Dopa (L)	DA (L)	DBS (L)	Duopa (L)
0	0.677966	1	0.75	0	1	1	1	0	1	0	0	0
1	0.423729	0	0.00	1	1	0	0	0	0	0	0	0
2	0.728814	0	0.25	0	1	0	0	0	1	0	0	0
3	0.694915	1	0.25	0	1	0	0	0	1	0	0	0
4	0.898305	0	0.25	1	1	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
99995	0.576271	1	0.50	0	1	0	0	0	1	0	0	0
99996	0.864407	1	0.75	0	1	0	0	0	1	0	1	0
99997	0.305085	0	0.25	0	1	0	0	0	1	1	0	0
99998	0.745763	1	0.75	1	1	1	0	0	0	0	0	0
99999	0.627119	0	0.25	0	1	0	0	0	1	0	0	0

Figures 36-37: Initial and normalized DataFrames from Prototype 3 with features (F) and labels (L)

Note that the hyperparameters were tuned prior to recording data in testing. The only changes made between versions were related to the architecture of the ANN (number of hidden layers, number of nodes, presence or absence of dropout regularization layer).

Also note that there is nearly zero variability in results when running the same model version simply due to the machine learning process remaining unchanged. Thus, results from only one trial of each version were collected, invalidating statistical tests.

Results

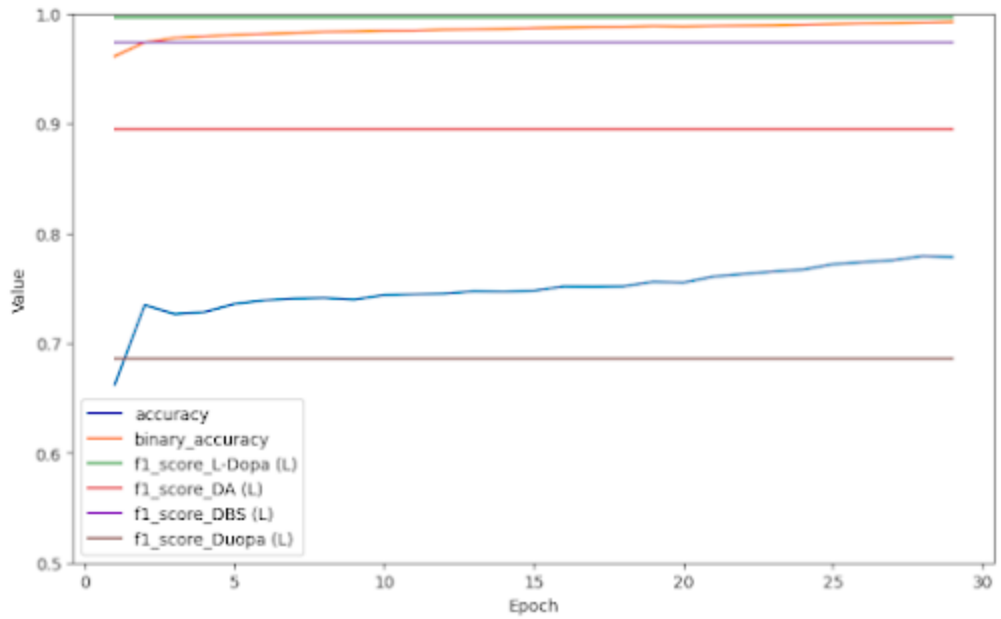
Version 1 (1 HL)

Model Specifics

- Architecture
    - Hidden Layers (HL): 1
        - First Layer Nodes: 32
        - Second Layer Nodes: N/A
    - Dropout Regularization Layer (DR): No
- Hyperparameters
    - Learning Rate: 0.01
    - Epochs: 30
    - Batch Size: 1000
    - Validation Split: 0.2

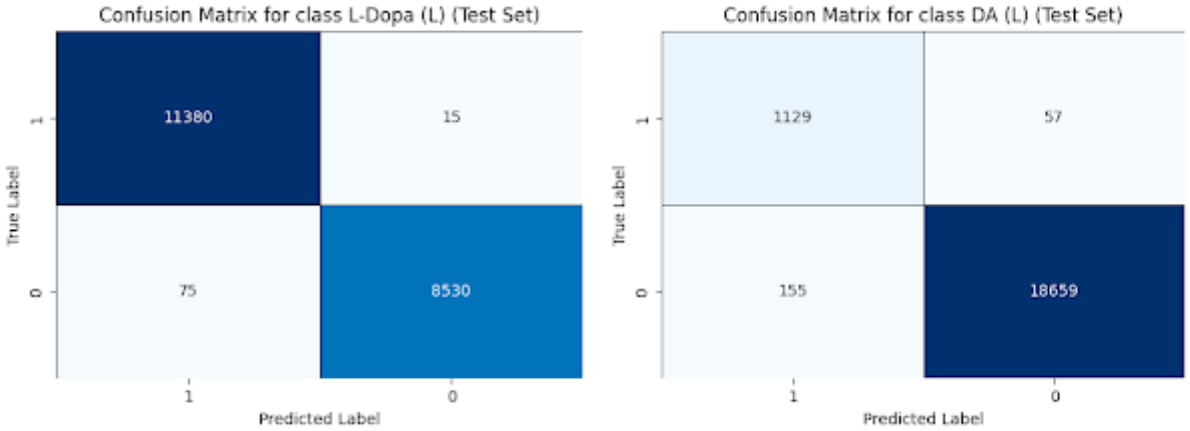
Performance Criteria

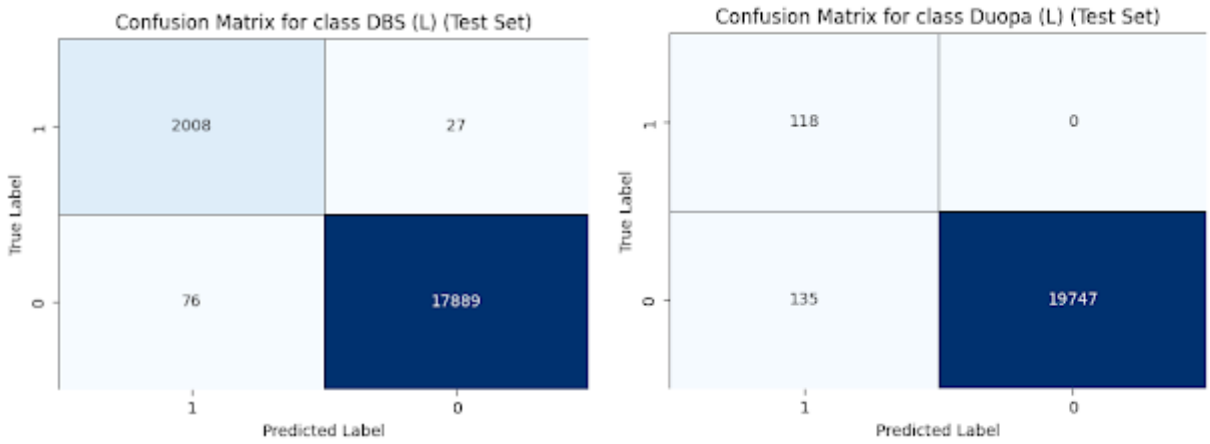
- Training and Validation Sets Evaluation
    - Macro-average F1 Score: 88.75%
    - Graph of Accuracy, Binary Accuracy, and Class F1 Scores over Epochs (shown below)



Graph 3.1: Accuracy, binary accuracy, and class F1 scores over epochs (Version 1)

- Test Set Evaluation
    - Loss: 1.9%
    - Accuracy: 79.00%
    - Binary Accuracy: 99.33%
    - Macro-average F1 Score: 88.03%
    - Confusion Matrices (shown below)





Confusion Matrices 3.1-3.4: Model predictions of each treatment class (Version 1)

- Observations
    - Accuracy increasing over epochs
    - Binary accuracy increasing initially then remaining relatively constant
    - When model predicts Yes (1) in test set
        - Levodopa: Often (57.3%)
        - DA: Infrequently (6.4%)
        - DBS: Infrequently (10.4%)
        - Duopa: Almost Never (1.3%)

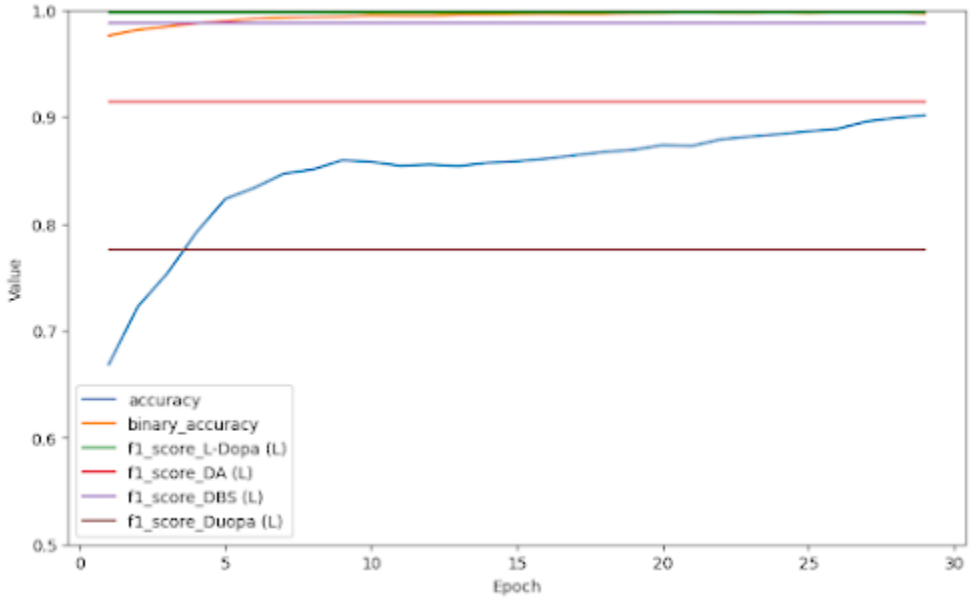
Version 2 (2 HL)

Model Specifics

- Architecture
    - Hidden Layers (HL): 2
        - First Layer Nodes: 32
        - Second Layer Nodes: 32
    - Dropout Regularization Layer (DR): No
- Hyperparameters
    - Learning Rate: 0.01
    - Epochs: 30
    - Batch Size: 1000
    - Validation Split: 0.2

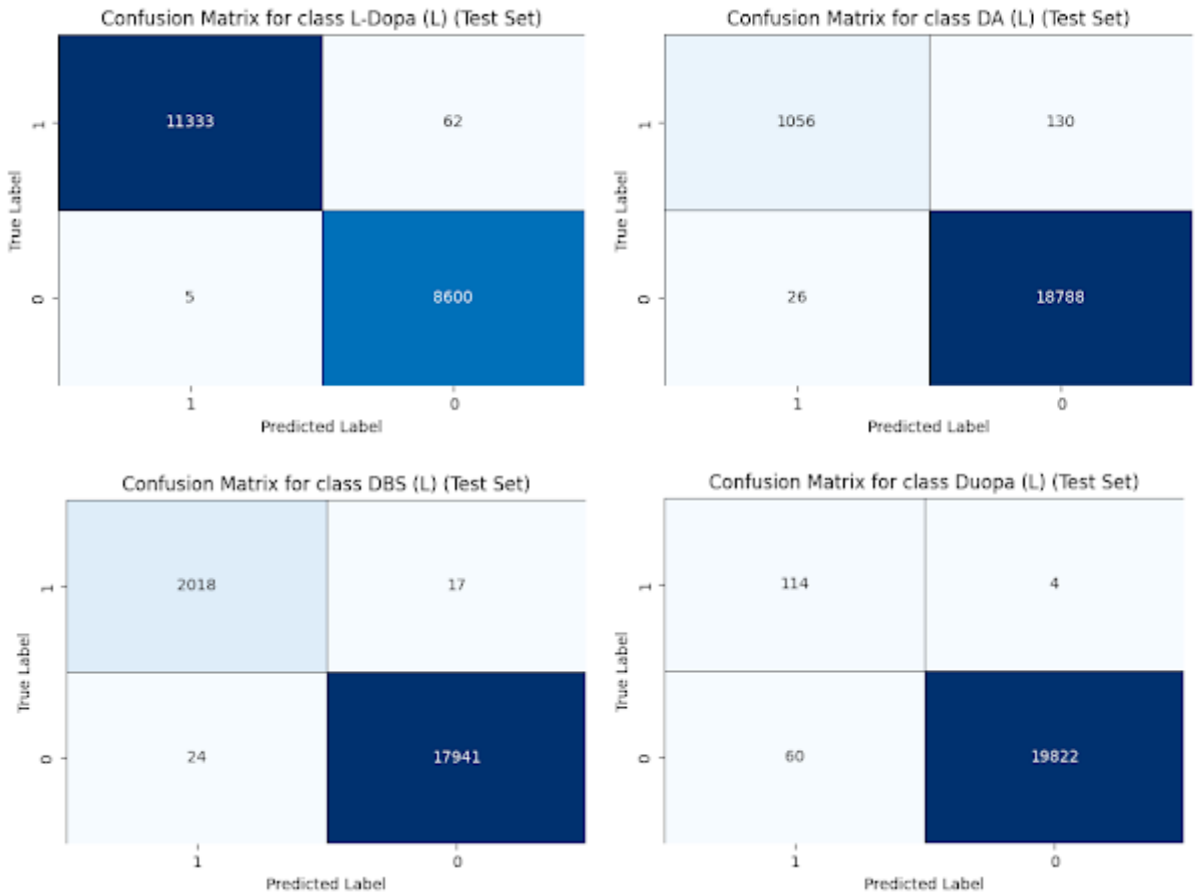
Performance Criteria

- Training and Validation Sets Evaluation
    - Macro-average F1 Score: 91.87%
    - Graph of Accuracy, Binary Accuracy, and Class F1 Scores over Epochs (shown below)



Graph 3.2: Accuracy, binary accuracy, and class F1 scores over epochs (Version 2)

- Test Set Evaluation
    - Loss: 1.02%
    - Accuracy: 87.35%
    - Binary Accuracy: 99.59%
    - Macro-average F1 Score: 92.48%
    - Confusion Matrices (shown below)



Confusion Matrices 3.5-3.8: Model predictions of each treatment class (Version 2)

- Observations
    - Accuracy increasing over epochs
    - Binary accuracy increasing initially then remaining relatively constant
    - When model predicts Yes (1) in test set
        - Levodopa: Often (56.7%)
        - DA: Infrequently (5.4%)
        - DBS: Infrequently (10.2%)
        - Duopa: Almost Never (0.9%)

Version 3 (2 HL, DR)

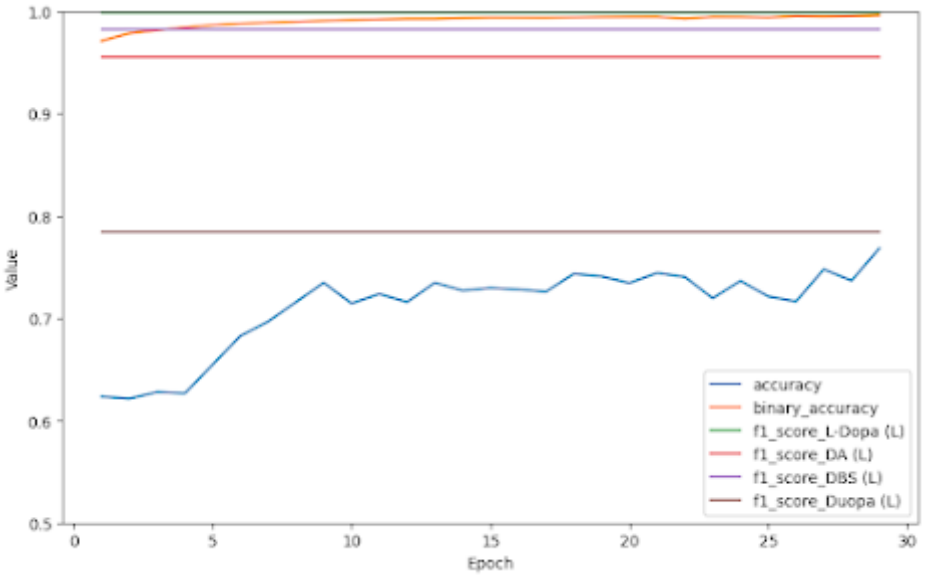
<u>Model Specifics</u>

- Architecture
    - Hidden Layers (HL): 2

        - First Layer Nodes: 32
        - Second Layer Nodes: 32
    - Dropout Regularization Layer (DR): Yes
- Hyperparameters
    - Learning Rate: 0.01
    - Epochs: 30
    - Batch Size: 1000
    - Validation Split: 0.2

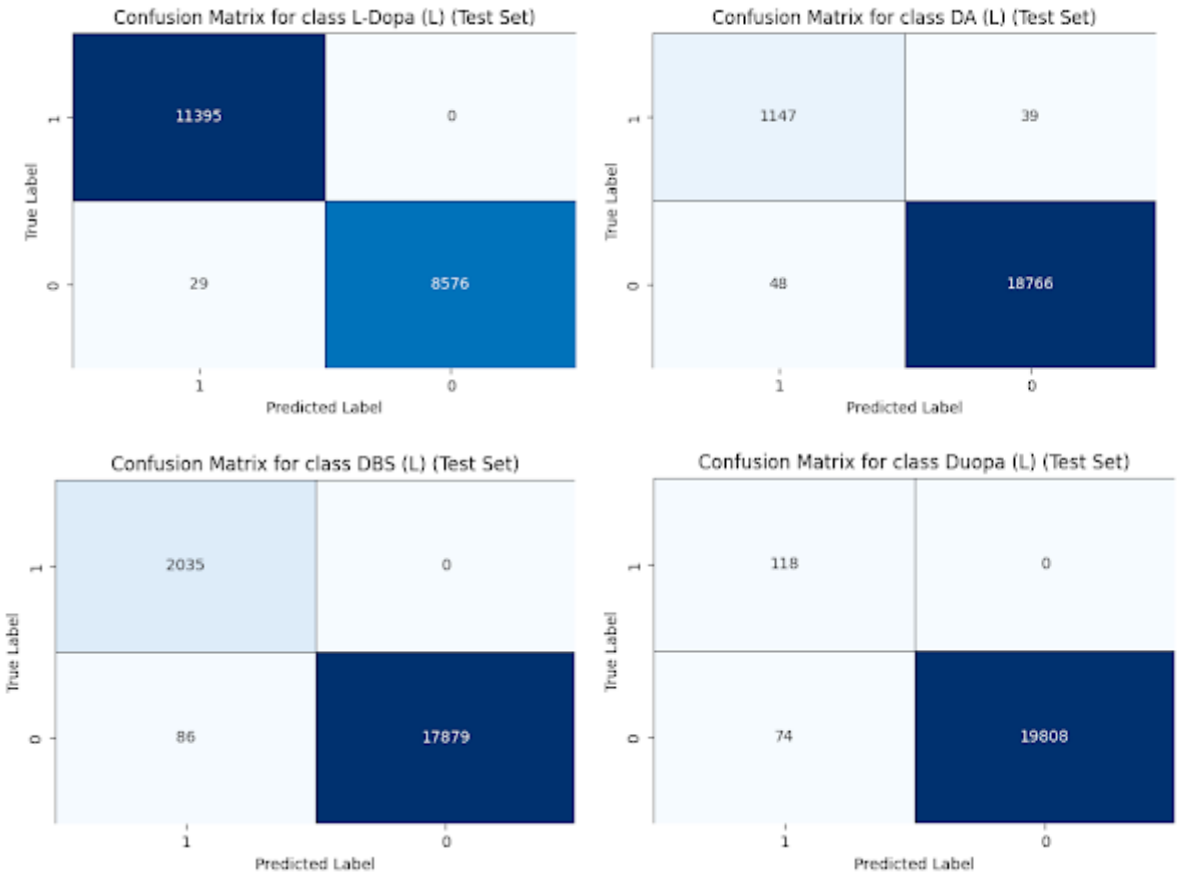
Performance Criteria

- Training and Validation Sets Evaluation
    - Macro-average F1 Score: 93.03%
    - Graph of Accuracy, Binary Accuracy, and Class F1 Scores over Epochs (shown below)



Graph 3.3: Accuracy, binary accuracy, and class F1 scores over epochs (Version 3)

- Test Set Evaluation
    - Loss: 0.07%
    - Accuracy: 79.34%
    - Binary Accuracy: 99.66%
    - Macro-average F1 Score: 92.57%
    - Confusion Matrices (shown below)



Confusion Matrices 3.9-3.12: Model predictions of each treatment class (Version 3)

- Observations
  - Accuracy increasing initially then fluctuating over epochs
  - Binary accuracy increasing initially then remaining relatively constant
  - When model predicts Yes (1) in test set
    - Levodopa: Often (57.1%)
    - DA: Infrequently (6.0%)
    - DBS: Infrequently (10.6%)
    - Duopa: Almost Never (1.0%)

Table 3.1: Summary of Prototype 2 Results (Test Set Evaluation)

Prototype 2 Results	Accuracy (%)	Binary Accuracy (%)	Average F1 Score (%)
Version 1 (1 HL)	79.00	99.33	88.03

Version 2 (2 HL)	87.35	99.59	92.48
Version 3 (2 HL, DR)	79.34	99.66	92.57

Table 3.2: Prototype Comparison of Optimal Versions

Prototype Comparison	Accuracy (%)	Binary Accuracy (%)	Average F1 Score (%)
Prototype 1 (Version 1)	94.40	88.54	21.07
Prototype 2 (Version 2)	79.05	82.98	21.48
Prototype 3 (Version 2)	87.35	99.59	92.48

Discussion

Based on these results, the most effective model was that of Version 2 (two hidden layers), with the highest average of accuracy and binary accuracy (93.47%). Version 3 (two hidden layers, dropout regularization layer) did have a slightly higher macro-average F1 score (92.57%), but since accuracy is the more prominent measure of model efficacy, Version 2 was deemed more optimal. Version 1 (one hidden layer) had the lowest accuracy and average F1-score, but remained similar in binary accuracy. The lack of depth with one hidden layer is probably decreasing the amount of complex processing and learning done by the neural network.

Fortunately, the prototype comparison table demonstrates that the adjustments to the input and output layers in Prototype 3 significantly improved binary accuracy and average F1 score compared to the previous models. The average of the accuracy metrics was also greatest for Prototype 3. Moreover, as the confusion matrices indicate, while the models in Prototypes 1 and 2 predicted No (0) every time for two classes, the model in Prototype 3 made Yes (1) and No (1) predictions for every class, being less affected by class imbalance. Overall, with an average

of 93.14% from the performance metrics in Table 3.2, Prototype 3 surpassed its predecessors and emerged the most effective model of this project.

In these tests, the example set of rules that influenced the optimal treatment predictions in Prototype 3 were quite concise and general. However, in application, the set of rules will be significantly more complex and specific, requiring thorough research in real world settings. Conceptually, this can be a process performed within a lab or hospital setting, with medical professionals creating an initial set of rules. After testing and modifying these rules over time, perhaps using UPDRS and MDS-UPDRS (Unified Parkinson's Disease Rating Scales), the effective ones can be translated to the neural network. At some point, the established set of rules paired alongside the predictive power of the machine learning model can assign optimal treatment plans for any patient with Parkinson’s disease.

Conclusion

This project aimed to create a revolutionary artificial neural network (ANN) model that accurately assigns the optimal treatment(s) for a patient with Parkinson's disease (PD) given multiple factors: age, sex, PD stage, and presence or absence of contraindications. Considering the high performance metrics of Prototype #3 particularly, the model was successful in accomplishing this goal.

However, in prototype progression, the application of this model altered slightly. The neural network began with hopes of being implemented directly with predictions from a simulated dataset. Upon testing with a realistic simulation dataset, this goal seemed rather impractical and somewhat ineffective due to constraints in dataset acquisition, class imbalance, and natural randomness. Rather, a new approach was formed in the last prototype that used the simulation dataset for the features and a set of rules for the labels, promoting primarily rules-based predictions.

This updated model has no effect in its current proof-of-concept stage. Eventually, after tests and modifications in real world settings, a functional, elaborate set of rules will be formed that can be processed by the neural network and applied to all patients. In order to progress this promising program, the following steps should be taken: utilize data of PD patients from hospitals directly if possible (for the features) and conduct research for the creation of the set of rules (for the labels).

Ultimately, the purpose of this machine learning model in application was to eradicate certain medication errors including contraindications as well as reduce pharmaceutical industry influence in prescription decision making. With refinement of the model and just implementation

in a hospital setting (impartial data collection and research, guidance for healthcare provider prescriptions, etc.), this purpose can be achieved.

References

Alshammri, R., Alharbi, G., Alharbi, E., & Almubark, I. (2023). Machine learning approaches to identify Parkinson's disease using voice signal features. *Frontiers in Artificial Intelligence*, 6, 1084001. https://doi.org/10.3389/frai.2023.1084001

Bajaj, A. (2021, May 2). *Performance Metrics in Machine Learning [Complete Guide]*. Neptune.ai. https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide

Bakker, M., Johnson, M. E., Corre, L., Mill, D. N., Li, X., Woodman, R. J., & Johnson, J. L. (2022). Identifying rates and risk factors for medication errors during hospitalization in the Australian Parkinson's disease population: A 3-year, multi-center study. *PLOS ONE*, 17(5), e0267969. https://doi.org/10.1371/journal.pone.0267969

Cao, Y., Kunaprayoon, D., Xu, J., & Ren, L. (2023). AI-assisted clinical decision making (CDM) for dose prescription in radiosurgery of brain metastases using three-path three-dimensional CNN. *Clinical and Translational Radiation Oncology*, 39, 100565. https://doi.org/10.1016/j.ctro.2022.100565

Davari, M., Khorasani, E., & Tigabu, B. M. (2018). Factors Influencing Prescribing Decisions of Physicians: A Review. *Ethiopian Journal of Health Sciences*, 28(6), 795–804. https://doi.org/10.4314/ejhs.v28i6.15

Deep Brain Stimulation (DBS) | *Parkinson's Foundation*. (n.d.). www.parkinson.org. https://www.parkinson.org/living-with-parkinsons/treatment/surgical-treatment-options/deep-brain-stimulation

Dopamine Agonists for Parkinson's Disease Treatment. (2017). *ParkinsonsDisease.net*. https://parkinsonsdisease.net/medications/dopamine-agonists

Draelos, R. (2019, February 23). *Measuring Performance: AUC (AUROC)*. Glass Box. https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

Duopa | *Parkinson's Foundation*. (n.d.). www.parkinson.org. https://www.parkinson.org/living-with-parkinsons/treatment/surgical-treatment-options/duopa

Elliott, C. (2014). Relationships between physicians and Pharma: Why physicians should not accept money from the pharmaceutical industry. *Neurology: Clinical Practice*, 4(2), 164–167. https://doi.org/10.1212/cpj.0000000000000012

Fickweiler, F., Fickweiler, W., & Urbach, E. (2017). Interactions between physicians and the pharmaceutical industry generally and sales representatives specifically and their association with physicians' attitudes and prescribing habits: a systematic review. *BMJ Open*, 7(9), e016408. https://doi.org/10.1136/bmjopen-2017-016408

Govindu, A., & Palwe, S. (2023). Early detection of Parkinson's disease using machine learning. *Procedia Computer Science*, 218, 249–261. https://doi.org/10.1016/j.procs.2023.01.007

Grissinger, M. (2018). Delayed Administration and Contraindicated Drugs Place Hospitalized Parkinson's Disease Patients at Risk. *Pharmacy and Therapeutics*, 43(1), 10–39. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5737245/

Houghton, R., Boess, F., Verselis, L., Ding, Y., Freitas, R., Constantinovici, N., & Ong, R. (2019). Treatment Patterns in Patients with Incident Parkinson's Disease in the United States. *Journal of Parkinson's Disease*, *9*(4), 749–759. https://doi.org/10.3233/jpd-191636

IBM. (2023). *What Are Neural Networks? | IBM*. Www.ibm.com; IBM. https://www.ibm.com/topics/neural-networks

Islam, S., Rahman, W., Abdelkader, A., Lee, S., Yang, P. T., Purks, J., Adams, J. L., Schneider, R. B., E. Ray Dorsey, & Hoque, E. (2023). Using AI to measure Parkinson's disease severity at home. *Npj Digital Medicine*, 6(1). https://doi.org/10.1038/s41746-023-00905-9

Jayaswal, V. (2020, September 8). *Performance Metrics: Accuracy*. Medium. https://towardsdatascience.com/performance-metrics-accuracy-118f728d328f

Jayaswal, V. (2020, September 15). *Performance Metrics: Confusion matrix, Precision, Recall, and F1 Score*. Medium. https://towardsdatascience.com/performance-metrics-confusion-matrix-precision-recall-and-f1-score-a8fe076a2262

K. Ray Chaudhuri, Kovács, N., Pontieri, F. E., Aldred, J., Bourgeois, P., Davis, T. L., Cubo, E., Marieta Anca-Herschkovitsch, Iansek, R., Siddiqui, M. S., Mihaela Simu, Bergmann, L., Ballina, M., Kukreja, P., Ladhani, O., Jia, J., & Standaert, D. G. (2023). Levodopa Carbidopa Intestinal Gel in Advanced Parkinson's Disease: DUOGLOBE

Final 3-Year Results. *Journal of Parkinson's Disease*, *13*(5), 769–783. https://doi.org/10.3233/jpd-225105

Karapinar Senturk, Z. (2020). Early diagnosis of Parkinson's disease using machine learning algorithms. *Medical Hypotheses*, 138, 109603. https://doi.org/10.1016/j.mehy.2020.109603

Larios Delgado, N., Usuyama, N., Hall, A. K., Hazen, R. J., Ma, M., Sahu, S., & Lundin, J. (2019). Fast and accurate medication identification. *Npj Digital Medicine*, 2(1). https://doi.org/10.1038/s41746-019-0086-0

Lertxundi, U., Isla, A., Solinís, M. A., Domingo-Echaburu, S., Hernandez, R., Peral-Aguirregoitia, J., Medrano, J., García-Moncó, J. C., (2017, March 1). Medication Errors in Parkinson's Disease Inpatients in the Basque Country. *Parkinsonism & Related Disorders*. https://pubmed.ncbi.nlm.nih.gov/28065403/

Lobo, A. (n.d.). *Older age at deep brain stimulation surgery for Parkinson's raises risks | Likelihood of dying in following years higher for patients in 60s, 70s | Parkinson's News Today*. Parkinsonsnewstoday.com. https://parkinsonsnewstoday.com/news/older-age-deep-brain-stimulation-parkinsons-death-risk/

Medical Advisory Secretariat. (2005). Deep brain stimulation for Parkinson's disease and other movement disorders: an evidence-based analysis. *Ontario Health Technology Assessment Series*, *5*(2), 1–56. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3382386/

Online Appendices. (2023). *CMAJ*. https://www.cmaj.ca/content/suppl/2016/05/24/cmaj.151179.DC1

Orozco, J. L., Valderrama-Chaparro, J. A., Pinilla-Monsalve, G. D., Molina-Echeverry, M. I., Pérez Castaño, A. M., Ariza-Araújo, Y., Prada, S. I., & Takeuchi, Y. (2020). Parkinson's disease prevalence, age distribution and staging in Colombia. *Neurology International*, *12*(1). https://doi.org/10.4081/ni.2020.8401

Prescription Medications | *Parkinson's Foundation*. (n.d.). www.parkinson.org. https://www.parkinson.org/living-with-parkinsons/treatment/prescription-medications

Pruneski, J., Williams, R. J., Nwachukwu, B. U., Ramkumar, P. N., Kiapour, A. M., Martin, R. M., Karlsson, J., & Ayoosh Pareek. (2022). The development and deployment

of machine learning models. 30(12), 3917–3923.
https://doi.org/10.1007/s00167-022-07155-4
Rizek, P., Kumar, N., & Jog, M. S. (2016). An Update on the Diagnosis and Treatment of Parkinson Disease. *Canadian Medical Association Journal*, *188*(16), 1157–1165.
https://doi.org/10.1503/cmaj.151179
Statistics | *Parkinson's Foundation*. (n.d.). www.parkinson.org.
https://www.parkinson.org/Understanding-Parkinsons/Statistics
Surgical Treatment Options | *Parkinson's Foundation*. (n.d.). www.parkinson.org.
https://www.parkinson.org/living-with-parkinsons/treatment/surgical-treatment-options#
Technologies, D. (2021, February 25). *Understanding the Confusion Matrix for Model Evaluation & Monitoring*. Datatron.
https://datatron.com/understanding-the-confusion-matrix-for-model-evaluation-monitoring/
Ting, H.-W., Chung, S.-L., Chen, C.-F., Chiu, H.-Y., & Hsieh, Y.-W. (2020). A drug identification model developed using deep learning technologies: experience of a medical center in Taiwan. *BMC Health Services Research*, 20(1).
https://doi.org/10.1186/s12913-020-05166-w
Willis, A. W., Roberts, E., Beck, J. C., Fiske, B., Ross, W., Savica, R., Van Den Eeden, S. K., Tanner, C. M., Marras, C., Alcalay, R., Schwarzschild, M., Racette, B., Chen, H., Church, T., Wilson, B., & Doria, J. M. (2022). Incidence of Parkinson disease in North America. *Npj Parkinson's Disease*, *8*(1). https://doi.org/10.1038/s41531-022-00410-y