The Promising Synthesis of Machine Learning, Python and Diagnostics

Summer S Will

August 22, 2021

Preface

Parkinson's Disease is a genetic disorder that has affected men in my family for generations, including my brother, father, and late grandfather. The disease is slow-moving but inexorably dreadful, leaving a person's conscious mind completely intact but attacking the brain's ability to, essentially, move.

My father is the perfect human being. He is perpetually kind, outrageously patient, and has provided me and my family with a high quality of life that we do not deserve. He, of course, has Parkinson's Disease and is moving towards its later stages. His unique demeanor has allowed him to deal with the disorder with a ridiculous amount of grace. He loves his family endlessly, and would continue to adore me no matter what I studied, but I have always been inclined to lean in his direction in some way during my education, whether it be studying something he wanted to study or studying something related to him.

As I move through my undergraduate education, and finally solidify the topics I will graduate having mastered, there is a distinct lack of anything medicine-related. I enjoyed the biological sciences in earlier education, but I find the most satisfaction with the final letter of the infamous STEM realm: mathematics. Specifically, statistics. Given these disciplines, a beginner might be hard-pressed to find absolute and immediate connections to Parkinson's Disease. This project is my personal exploration of the effort to connect mathematics and solutions to Parkinson's. As the reader will learn, Parkinson's has no cure and affects the brain in drastic ways long before it is able to be diagnosed. The idea, therefore, is to diagnose as early as possible. I feel beyond grateful to have learned, through this project, one way that a statistician may assist in the earliest possible diagnosis of Parkinson's Disease.

The reader should not expect any further waxing poetic. A strictly informational branching of computer science and statistics to diagnostics is not the most delicate ode to family, but I think my own family would expect nothing less.

This one's to you, Dad.

The dream of greatness is the infallible proof of mediocrity, because that sort of world is what the man of achievement would not be able to bear.

- Ayn Rand

Introduction

machine learning: An Extension of Statistical Learning

Statistical learning provides the framework for machine learning, but the two generally have different focuses and abilities. Statistical learning is a recently-developed field of mathematics that surrounds analyzing data. It provides the basis for both inferring relationships between variables and predicting outcomes based on variables, but it focuses on the former. machine learning utilizes the same principles developed in statistical learning, but focuses on predicting outcomes from a given data set. Both sciences are data-focused, obviously, but statistical learning relies on orders and detailed instruction from an outside source while machine learning has been created to be able to work on its own: interpolating data, finding outcomes, and fitting models independently. In addition, statistical learning is better used with small amounts of data sets–small samples, amounts of variables, etc.–while machine learning can work with billions of variables. There are more sophisticated similarities and differences between the two fields, but they will be explicitly discussed here. This project will provide descriptions of the most important aspects of the statistical learning framework as a foundation for machine learning later on.

About the Data

Advertising and Sales Budget Data

The statistical learning portion of this project borrows significantly from the wonderful textbook *Introduction to Statistical Learning* by Gareth James in the way of structure, notation, and analogy. The web-based data set that will be carried throughout as a means of motivating our study of statistical learning is based on a theoretical company's advertisement budgets and resulting sales. [2]

Parkinson's Speech Analysis Data

Parkinson's disease is a genetic disorder of the nervous system that attacks the brain's ability to produce dopamine. People with the disease experience various speeds of declining physical health, with symptoms that can include but are not limited to tremors in the limbs, slowed speech, inability to use facial muscles, and shuffled steps. It has no cure, and has progressed extensively by the time symptoms appear. The name of the game, therefore, is to diagnose the disease as early as possible so as to begin medical treatment to slow decline. The disease will be discussed in more detail later on.

The Parkinson's data used throughout the machine learning section of this project was created by Max Little of the University of Oxford, in collaboration with the National Centre for Voice and Speech, in Denver Colorado, who recorded the speech signals. The original study published the feature extraction methods for general voice disorders. [3]

Extensive comprehension of this particular data set, including its variables and precisely how those variables were measured or calculated, is not the focus of this project. Provided are brief descriptions of each variable and the number of observations and variables. Later, we will apply machine learning to make predictions about the data and measure the accuracy of the project as a whole for further use.

This data set is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's Disease (PD). Each column in the table is a particular voice measure, and each row corresponds to one of 195 voice recordings from these individuals. The main aim of the data is to discriminate healthy people from those with PD. The health status of each individual is organized in the tables by a column set to 0 for healthy people and 1 for PD. The health status is our *outcome*, our Y variable that we will set aside when initially making predictions and then use as a measure for calculating the accuracy of our model. [4]

This Project

Purpose and Audience

The purpose of this project is to bridge the gap between statistics, computer science, and a small aspect of diagnostics. The intended audience can be a participant in any one of those fields, in any capacity, looking to branch towards another of the involved fields. Readers who are students of statistics may be looking to start working with Python–this project is one way to start. Readers who are students of computer science will find the Python instruction rudimentary but may be interested in learning a quick application of machine learning.

Organization

All topics covered in this project aim to bolster the reader's understanding of the act of using Python to conduct machine learning and detect Parkinson's disease among a sample of patients. In order to carry out this aim, we first describe some general aspects of statistical learning that provide a framework for machine learning. We then conduct the crux of the project, using Python to detect Parkinson's among the patients in our data set. Finally, we will discuss further applications for the tools used in Parkinson's detection, including use of the Python libraries and execution of machine learning.

Notation

In addition to the structure and data used in the statistical learning portion of this project, we will utilize the style of notation found in Introduction to Statistical Learning (ISL). Blocked below is James's//the ISL authors' summary of some notation:

> Let x_{ij} represent the value of the jth predictor, or input, for observation i, where $i = 1,2,...,n$ and $j = 1,2,...,p$. Correspondingly, let y_i represent the response variable for the ith observation. Then our data consist of $(x_1,y_1),(x_2,y_2),...,(x_n,y_n)$ where $x_i = (x_{i1},x_{i2},...,x_{ip})$.

In other words, we use n to denote the number of observations, p to denote the number of variables (henceforth referred to as *predictors*), i to denote a one general observation out of the n observations, and j to denote one general predictor out of the p predictors. In addition, the response variable (henceforth referred to as the *outcome*) for observation i is denoted by y_i.

Acknowledgements

First and foremost, thank you to my advisor, Professor Jo Hardin, whose patience and assistance have been instrumental in this project's creation. Thank you in addition to the committees behind the Remote Alternative Independent Summer Experience program for 2021; I am grateful every day to be part of an institution such as Pomona College, one which places the highest importance on community engagement and provides plentiful resources for the intellectual stimulation of its students in summers of quarantine. Special thanks to my brilliant friends, Carl Bell, Lucas Tiangco, and Nam Do for their continued assistance and support throughout both this project and my life.

Statistical Learning

Let's begin with an example. [2] Say we are tasked with deciding the optimal amount of money a company with a certain product should put into its advertisements. The company has three types of advertising it utilizes: newspaper, radio, and TV. We want to find out the optimal funds to put into each type in order to produce the largest amount of product sales. Translating this example into pieces of a theoretical data set, we will say that there are: n recordings of advertising funds, our *observations*, given by X; p types of advertising, our *predictors*, for each observation; and n recordings of product sales related to each observation of advertisement funding, given by Y.

Purpose

Now that we have established some observations, predictors, and outcomes, we can decide exactly what we want to do with them. Given a data set with n observations and p predictors, we may wish to find a relationship between the observations and outcomes through a function f or we may wish to find the outcome itself, y. The former aim is referred to as $inference$, while the latter as *prediction*. As is often the case with medical studies, in this project we wish to predict the outcome Y, as we do not particularly need to understand the relationship between individual predictors for each patient and the outcome as much as we desire the outcome itself: a Parkinson's diagnosis or not. Another way to differentiate $inference$ and *prediction* is to consider whether or not our predictors can change, or if it is our goal to change predictors. If the predictors are inherent medical qualities that will exist whether the study is conducted or not, then we are predicting. If, however, we were working with data for a different purpose, we may require inference. For example, to figure out the efficacy of different advertising types for an advertising firm on a specific population, we would want to know the relationship between the types of advertising and their efficacy. In this case, the predictors can and will change. Changing the predictors accordingly is the core purpose of a study in advertising.

Methods

Continuing with our analysis of prediction vs. inference, we must choose the correct model to fit our aim. Nature rarely produces perfect relationships between predictors and outcomes, so we can choose fits that have a range of closeness with matching the data. Functions lie on a spectrum of flexibility. A function is highly flexible if it matches each data point closely, giving a jagged shape. A function is less flexible the more it smooths out, generally connecting each data point. Some functions can be smooth in nature if the relationship between the data points is smooth in nature as well, but natural relationships are rarely smooth enough to produce an inflexible graph while also matching points. Figure 1.1 shows varying fits for a certain collection of data. Figure 1.2 shows the same fit applied to different samples of data.

Regression Versus Classification Problems

So far we have focused on regression, where the inputs and outputs are all quantitative; they can be placed on a scale of numbers with any amount of specificity required. While situations involving qualitative inputs are considered regression, classification deals with qualitative outcomes. Later in this project we will be working strictly with classification, but discussion of regression is necessary in order to understand the statistical framework involved in machine learning.

How Accurate Is the Model?

Our data comes in *samples* because we cannot take observations from every part of a population. We take multiple samples in order to make our end results more accurate. Despite the samples having ample observations and coming from the same population, we will see different results in each sample we draw. When we *fit* models to our data, whether we are predicting or inferring, we must aim for the most accurate fits possible. There are ways to measure the accuracy of our predictions, and ways to do so for both regression and classification are described here.

Measuring the Quality of Fit: Regression

With regression, we measure the accuracy of the model with *mean squared error* (MSE). As can be seen in the given formula below, the MSE takes the sample outcome value, subtracts it by the estimated outcome value, squares that quantity, takes a sum of this quantity for each observation, and divides the sum by n.

$$\text{MSE} = \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$$

Minimizing the MSE is one way to reach as accurate of an estimate as possible for a given set of data.

Bias and Variance

Two measures we use to gauge the accuracy of our samples are *bias* and *variance*. Say we have two variables in a sample that form a near-perfect linear relationship. If we were to take two different samples, one with a wayward variable, one that does not fall into the linear relationship, the inferences we come to for the two samples would not be much different. This indicates low variance. On the other hand, if we have a relationship that is not strongly linear and our model matches each data point pretty closely, a change in sample will cause a drastic change in variance. On the other hand, given the former example, changing the fit to be a different line will cause a drastic change in how the data points suit the model. This creates high bias, whereas changing the model a small amount in the latter example will not change how the variables relate to the model very much–low bias.

Measuring the Quality of Fit: Classification

The MSE can be applied to a wide range of prediction or inference problems, but a better way to measure accuracy of a classified prediction is through a *K-nearest neighbors* (KNN) algorithm:

> Given a positive integer K and a test observation x_0, the KNN classifier first identifies the K points in the training data that are closest to x_0, represented by N_0. It then estimates the conditional probability for class j as the fraction of points in N_0 whose response values equal j:
>
> $$\Pr(Y = j|X = x_0) = \tfrac{1}{K} \sum_{i \epsilon \mathcal{N}_0} I(y_i = j).$$ [2]

In other words, the KNN classifier uses a certain amount of training data (K) predictors nearest to out test data predictor, then estimates the probability that the outcome will fall within the range of outcomes resulting from the K training data predictors.

Machine Learning

We have already discussed ways to make predictions or inferences based on data, and how to use regression analysis to do so for quantitative predictors and outcomes. In addition, we have gone over the possible extension of these methods to classification, in which qualitative prodictions are made. Here we will go over the specific process of finding the methods and conjuring the models for detecting Parkinson's disease. Since the given Parkinson's data involves about six thousand numbers, a higher functioning method of prediction is required for the most accurate results. We now move on from Statistical Learning to machine learning!

A Prophetic Discrepancy from Statistical Learning

machine learning, in general, is a technique of predicting the future or classifying information in order to find an answer to a proposed task; it utilizes algorithms that have been previously trained by a user via data sets. These algorithms are able to "learn from past experience and analyze the historical data". [] The ultimate goal of machine learning is to develop "intelligent systems that are capable of [making] decisions on an autonomous basis". []

It can be difficult to immediately reconcile machine learning with Statistical Learning because of the former's novel characteristic of *autonomy*. Where does the autonomous aspect come in with respect to a more "involved" approach to data? Why is machine learning any different from making predictions with Statistical Learning methods? machine learning naturally extends from Statistical Learning methods. They are the same, essentially, except for when massive amounts of data are necessary and prediction is the end goal, what was once Statistical Learning is now machine learning. The difference is almost exclusively semantic, and the implementation of machine learning would not be so groundbreaking if not for the flourishing utilization of Artificial Intelligence, which is to machine learning as machine learning is to Statistical Learning: an extension. One almost gets the sense that Statistical Learning had a particular *advancement* in the way of larger amounts of data and a necessity to predict from that data, but after Artificial Intelligence came into play, such a *advancement* was retroactively named machine learning.

Decision Trees

We will describe decision trees via example. There are twenty-two predictors ("attributes") involved in the Parkinson's speech analysis data. For simplicity, we will describe five of them here (and name them in ways that are not necessarily accurate to their true meanings): waver, speed, frequency, shimmer, amplitude. Each will be measured on a scale of 0 to 100, 0 being low waver, speed etc., and 100 being high. Starting with waver, we'll split the observations into two groups, those with less than 50 waver and those with equal to or more than 50 waver. Taking those observations with ¡50 waver, we'll further split them into those with ¡50 shimmer and ¿=50 shimmer. Each specification, e.g. waver ¡ or ¿= 50–, is called a *node*. Nodes that do not further split into two more nodes are called *terminal nodes*. Nodes than do split are called *internal* nodes.

One collection of nodes, internal nodes, and terminal nodes–connected by "branches"–is a decision tree. Considering different variables placed in each position, the creation of an inconceivable amount of decision trees is possible for large data sets. This is why decision trees are often used in machine learning, where a data set can simply be provided and machine learning algorithms will be able find patterns and make predictions autonomously.

(Gradient) Boosting

Gradient boosting is "a machine learning technique for regression and classification problems that produces a prediction mode in the form of an ensemble of weak prediction models". [1] Hence, the term "boosting", i.e. "making more robust" by using the combined power of many weak predictions. Many prediction models can be made more robust by gradient boosting, including decision trees, which is the central classification method for this project.

Detecting Parkinson's Disease

We made it! Let's begin the process of detecting Parkinson's disease through Python's machine learning capabilities.

We're going to walk through the process step-by-step and explain what each act is accomplishing based on our knowledge of statistical learning and machine learning. Whereas our previous explanations of topics were somewhat terse, these will be comprehensive [5]. The following steps assume the reader has downloaded Python's latest version. In addition, these steps are directly applicable for computers using MacOS; the same code is not provided for PC, but is easily translatable.

1. **Download Python Libraries**. Within your computer's command line (PC) or Terminal (Mac), install necessary libraries with the following code:

   ```
   pip install numpy pandas sklearn xgboost
   ```

2. **Install Jupyter Lab**. Jupyter lab is your integrated development environment (IDE), the tool through which you will use Python. Python does not showcase itself on a device directly, it simply works in the background of various other locations; some environments (like IDEs) are created specifically for the purpose of streamlining programming languages such as Python. Jupyter lab is particularly nice for this project because it will house all of your actions in one "notebook", allowing you to look back at mistakes or successes without losing track of your progress.

3. **Run Jupyter Lab**. Simply type the following code into Terminal to open a Jupyterlab tab in your browser.

   ```
   Jupyter lab
   ```

   Create a new console for your project code. Type code in the bottom of the console and use Shift+Enter to run the code.

4. **Import Python Libraries and Tools**. "Libraries" are open-source downloadable or pre-downloaded collections of reusable code. Python comes with hundreds of libraries and tens of thousands more exist to be downloaded for certain purposes. This project utilizes this ability, requiring the download of several Python libraries that will assist with certain machine learning tasks: numpy, to support a large collection of data; pandas, for data manipulation and analysis; scikit-learn, which contains the appropriate classification algorithms; and xgboost for carrying out gradient boosting.

```
import numpy as np
import pandas as pd
import os, sys
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

5. **Read the Data**. To “read data” is simple in theory but can be complicated in practice because of difficulties in knowing your file’s location and being able to correctly tell the program how to find it.

   First, install the following data (described earlier in the “Parkinson’s Speech Analysis Data” section): parkinsons.data. For clarity, lick on the “parkinsons.data” link you see on the link provided. When prompted to open or download, download using the default “TextEdit” format, not Excel or another format.

   Next, make sure you know exactly where your file has been saved; it should show up in your downloads folder. Python looks for files in your computer in a highly systematic and specific way: the location of the file is called its *path*, indicating its “path” from the file itself backwards in more and more general locations. In order to find the file’s path, return to Terminal (you may need to restart Terminal in order to clear your previous work on it). For simplicity, we will use the method of finding our file’s path by dragging and dropping it into Terminal. So, go to your downloads through Finder, locate “parkinsons.data”, and drag and drop it into Terminal. Terminal should come up with highlighted text that looks something like this:

   ```
   /Users/loan/downloads/parkinsons.data
   ```

   The most general path location is “Users”, followed by the user herself, then the folder that contains your file, and finally the file name. My user name is “loan”. Regardless, Terminal will tell you exactly where your file is.

   Now, to actually read the data in Jupyterlab, type the following command:

   ```
   df=pd.read_csv(r"/Users/loan/downloads/parkinsons.data")
   df.head()
   ```

These commands will pull up the first five rows of your data. Important note: do not type exactly "/Users/loan/downloads/parkinsons.data" where you see it in the provided code, but type whatever output Terminal gave you when you dragged and dropped the Parkinson's data into it.

6. **Specify Predictions and Labels**. The raw parkinsons.data is only a series of letters and numbers separated by tactful commas; we need to tell Python which parts of the data are predictors and which are simply labels (which person is being tested and what the health status of each person is).

   ```
   features=df.loc[:,df.columns!='status'].values[:,1:]
   labels=df.loc[:,'status'].values
   ```

7. **Counting Each Status**. As described above, the health status for each individual (a categorical variable) is numerically coded with zeros to indicate no PD and ones to indicate PD. Let's test Python's understanding of our data to get the count of each type of label in the status column.

   ```
   print(labels[labels==1].shape[0],
   labels[labels==0].shape[0])
   ```

   We should have 147 ones and 48 zeros in the status column of our dataset.

8. **Scale the Predictors**. Currently, the numerical values of our predictors are not scaled so that we can compare them in any meaningful capacity. Let's scale them to between -1 and 1 to normalize them. The MinMaxScaler transforms features by scaling them to a given range. The fit_transform method fits to the data and then transforms it.

   ```
   scaler=MinMaxScaler((-1,1))
   x=scaler.fit_transform(features)
   y=labels
   ```

9. **Split the Data Set**. We are provided with this data set that already gives the outcomes we wish to predict. Our aim is to create a model that can accurately detect Parkinson's disease in an individual, which of course must happen before we are aware whether or not this person has Parkinson's disease. Splitting the data set gives us the opportunity to use one part to create a model and another part to test the accuracy of our data. As descibed above, the former data will be treated as our *training* data (because we use it to *train* a working model) and the latter data will be treated as our *test* data. We will keep 20 percent of the data for testing.

```
x_train,x_test,y_train,y_test=train_test_split(x, y,
test_size=0.2, random_state=7)
```

10. **Build the Model**. We now carry out the most significant aspect of this project: building the best prediction model for our data set. As described earlier with regard to more classical practices–like regression–our aim is to find a way to organize our data in some way as to produce an accurate prediction of an outcome. Here, our goal is not to create a deep understanding of the relationship between predictors and outcomes. We instead wish to use the given predictor values to predict outcomes for each observation. One way to do this for classification is via *decision trees*.

    ```
    model=XGBClassifier()
    model.fit(x_train,y_train)
    ```

11. **Calculate the Accuracy**. Finally, we will generate predicted values for the test data and calculate the accuracy of our model. We are able to do this because we know the true outcomes, so we can measure our predicted outcomes against them.

    ```
    y_pred=model.predict(x_test)
    print(accuracy_score(y_test, y_pred)*100)
    ```

Summary

The ability to take a sample of patients' attributes and accurately detect Parkinson's disease is nothing short of groundbreaking.

Afterword: For Beginners

When it comes to learning Python, you will learn to walk after you start running. I started this project by seeing the outcome of the project: DataFlair's showcasing of the XGBoost capabilities. I thought at the time I would need to work very hard to work my way up to the Python competence required to carry out the tasks. I saw words like "classifier", "boosting", and "decision trees" and wrote them off as things I could learn with a quick research. It turns out, I had the learning strategies flipped. In order to learn Python, you must simply *begin*. Given a project, try to do it; at each crossroad or dilemma, find help on how to move past it, whether that help is from the internet or a fellow student. In order to learn statistics, however, especially the kind of statistics I was observing in action, I recommend skipping the least details possible. For instance, I was looking at a process that involved a fairly advanced (but not complicated!) form of classification: decision trees. In order to carry out decision tree classification from scratch, one must learn about the classical classification methods: linear regression and linear discriminant analysis. In order to understand classical classification methods, one must first understand classical regression methods. Overall, the connection between these fields in the backwards way just described can be summarized by learning statistical learning theory and *then* machine learning. Clearly I had underestimated the scope of statistics I was looking at and overestimated the Python skill required in order to simply begin a project.

In short, if you feel cheated on your understanding of statistical learning based on this paper, go shore up your misunderstandings. I did my best, but now you must do yours.

Moreover, do not waste time learning each and every bit of Python you can before actually beginning a task or specific project. It's not worth your time, since most of the rudimentary commands you learn in some tutorial will fall out of your brain quickly. Attaching your understanding of the concepts Python projects to the code required is the only way to solidify that code.

References

[1] DataFlair. “gradient boosting algorithm - working and improvements.”, 2021.

[2] Gareth James, Daniela Witten, Robert Tibshirani, and Trevor Hastie. *Introduction to Statistical Learning.* Springer Texts in Statistics, New York, NY, 2013.

[3] Max A. Little, Patrick E. McSharry, Eric J. Hunter, and Lorraine O. Ramig. Suitability of dysphonia measurements for telemonitoring of parkinson’s disease, 2008.

[4] Little MA, McSharry PE, Roberts SJ, Costello DAE, and Moroz IM. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection, 2007.

[5] R. Muller. *Energy for Future Presidents: The Science Behind the Headlines.* W.W. Norton, New York, NY, 2012.