# Machine learning alternatives for the diagnosis of ADHD from functional connectivity and phenotypic information

Amrit Baveja
May 2018

Codebase DOI: **https://doi.org/10.5281/zenodo.1227447**

GitHub Repository accompanying this paper: **https://github.com/abaveja313/bsrp**

# Introduction

ADHD overdiagnosis is an unfortunate and known problem in our society. Currently, ADHD is diagnosed from surveys and conversations, which are up to the interpretation of the doctor. [School Mental Health Project, n.d.]. In fact, ADHD is something that organizations push as "self-diagnosable": "General and social networking media and ADHD organizations are seen as contributing to increasing demand, overdiagnosis, and over prescription. For example, a variety of websites promote initial home-diagnosis based on a small set of symptom-related questions" [School Mental Health Project, n.d.]

Additionally, as the School Mental Health Project says, a lot of doctors tend to focus very little on evidence refuting a positive ADHD diagnosis and only acknowledge evidence that supports their claim: "Confirmation bias suggests the tendency for one to become attached to an unconfirmed hypothesis and subsequently only attend to and seek information that support the hypothesis while disregarding and minimizing counter-evidence" [School Mental Health Project, n.d.]. Furthermore, a very small percentage of doctors actually use multiple sources in deciding a diagnosis: 15%. [School Mental Health Project, n.d.]. Eighty five percent of patients are being diagnosed with only one or two sources, so it is probable that evidence pointing to a negative diagnosis is being suppressed.

Confirmation bias is not the only type of partiality that exists during the ADHD diagnosis process. In an interview with the Los Angeles Times, Richard M. Scheffler, a health economist from UC Berkeley, stated that the current process for diagnosing ADHD depends heavily on biased judgments from parents and clinicians. [School Mental Health Project, n.d.]. Parents, although they often provide insight into how the child acts and focuses at home, can often use medication as an easy way out. ADHD medication causes children to be more mellow and more attentive-- allowing them, in a lot cases, to be easier to handle. However, the problem is compounded by the doctors themselves. It is beneficial for the doctors, pharmaceutical companies, and to pharmaceutical distributors to heavily market these drugs because more sales, regardless of whether or not the patient actually needs them, brings them more revenue. [School Mental Health Project, n.d.] Also, pharmaceutical companies don't just push the drugs onto the doctors and parents, they market them to the consumers themselves[School Mental Health Project, n.d.]

However bleak ADHD diagnosis seems, there are encouraging signs as well. A study in the Netherlands found convincingly that ADHD is not something that is created by society; rather it is a real brain disorder. The scientists in this study found this by using a very large sample of ADHD/Non ADHD (control group) patients, "Our sample comprised 1713 participants with ADHD and 1529 controls from 23 sites with a median age of 14 years (range

4–63 years).” (Hoogman et. al, 2017). Dr. Hoogman and her team found a difference in size of the brain between certain ADHD and non-ADHD subjects: “The volumes of the accumbens (Cohen's d=−0·15), amygdala (d=−0·19), caudate (d=−0·11), hippocampus (d=−0·11), putamen (d=−0·14), and intracranial volume (d=−0·10) were smaller in individuals with ADHD compared with controls in the mega-analysis. There was no difference in volume size in the pallidum (p=0.95) and thalamus (p=0·39) between people with ADHD and controls. " (Hoogman et. al, 2017). The clear relationship between brain anatomy and ADHD is very encouraging as biological evidence which could be used in the future for ADHD diagnosis.

In addition, to using brain anatomy, computerized diagnosis of diseases has become more and more prevalent in the medical profession. Two extremely powerful tools for doing this are MRI scans and machine learning.

An MRI scanner, like an X-ray or a CT scanner, is used to create digital renderings of the human body. [Notter, 2016] In quantum mechanics and particle physics, every particle has a property called spin, which is decided randomly. The MRI’s powerful magnet (the average MRI scanner has a strength of 0.5 to 3 tesla, while a household magnet has 0.001 tesla [Monks, 2017] [HowStuffWorks, 2001]) is able to align particles forming the target part of the body from their random spin into one uniform direction. [Notter, 2016] Then, a sequence of radiofrequency (RF) pulses are fired at those particles across the magnetic field created by the scanner and the protons get aligned with the RF pulse and spin at the same time and in phase. [Notter, 2016]. When the pulse is removed and then fired again, the protons fall back to their initial state and realign with the magnetic field at different rates (depending on the type of tissue) and the energy that they give off can be measured with the MRI scanner. [Notter, 2016] The release of energy can be used by the MRI scanner to create a 3D/4D image of the brain. [Notter, 2016]

There are three types of MRI scans: Structural Magnetic Resonance Imaging (sMRI), Functional Magnetic Resonance Imaging (fMRI) and Diffusion Weighted Magnetic Resonance Imaging (DW-MRI) [Notter, 2016]:

- Structural Magnetic Resonance Imaging (sMRI) is a category of MRI Imaging that generates images by measuring the amount of water in a given location. [Notter, 2016]. Through this process, sMRI is able to create a detailed model of the target region by measuring water amounts in 2D slices, which are pieced together to form a complete 3D image. [Notter, 2016]. It is important to note that the anatomy of the brain is not supposed to change during acquisition of the sMRI. [Notter, 2016].

- Functional Magnetic Resonance Imaging (fMRI) is one of the most common categories of MRI scans for neuroscience. Rather than creating 3D renderings of the target region, like sMRI does, it creates 4D models of the target region—3D

images over time. fMRI is most commonly used in the brain for measuring neural activity. [Notter, 2016]. It works by detecting variations in blood oxygen level that occur over time series, sometimes as a response to external stimulation (like pictures or sounds). [Notter, 2016]. If stimulation is present, the location in the brain where the most neural activity is taking place will have increased blood oxygen levels. [Notter, 2016]. Following the increase of $O_2$ the blood oxygen level will recede because of local consumption at the region where the neural activity is present. [Notter, 2016]. As a result, a new flow of oxygenated blood will flow to this location. [Notter, 2016]. "After 4-6 seconds, a peak of blood oxygen level is reached. After no further neuronal activation takes place the signal decreases again and typically undershoots, before rising again to the baseline level." [Notter, 2016]. fMRI scans measure these variations in blood oxygen levels at specified regions-- the changes in the magnetic field are caused by differences between the magnetic susceptibility between oxygenated and deoxygenated blood. [Notter, 2016] This signal is called the Blood Oxygen Level Dependent (BOLD) response. [Notter, 2016].

There are three different kinds stimuli designs or paradigms for fMRIs: event related, block design and resting state. "Event-related means that stimuli are administered to the subjects in the scanner for a short period and in a random order. Stimuli are typically visual, but auditory and/or other sensory stimuli could also be used." [Notter, 2016]. "If multiple stimuli of a similar nature are shown in a block, or phase, of 10-30 seconds, that is a block design." [Notter, 2016]. "Resting-state designs acquire data in the absence of stimulation. Subjects are asked to lay still and rest in the scanner without falling asleep. The goal of such a scan is to record brain activation in the absence of an external task. This is sometimes done to analyze the functional connectivity of the brain." [Notter, 2016]

- The last type of MRI scan is called Diffusion Weighted Magnetic Resonance Imaging (DW-MRI). DW-MRI works by recording the trajectory of molecules in a given voxel (3D pixel). [Notter, 2016]. By doing this, one can make inferences about the structure of the underlying voxel. "For example, if one voxel contains mostly horizontal fiber tracts, the water molecules in this region will mostly diffuse (move) in a horizontal manner, as they can't move vertically because of this neural barrier." [Notter, 2016]. Diffusion MRI is used to gain information about the brain's white matter connections. [Notter, 2016].

However, in order to do statistical analysis or create a predictive model using an MRI volume, we have to account for fluctuations (such as the time between slices and head

movement) in the MRI data itself. To combat this, we use a strategy called preprocessing on the volumes to correct and unify them. The most common kinds of preprocessing are slime timing correction, motion correction, artifact detection, coregistration, normalization and smoothing. [Notter, 2016]:

- Slice timing correction (fMRI only) is used to correct missing values in between the 2D slices. [Notter, 2016]. For example, an fMRI with a slice time (the number of seconds between each slice is generated) of one second, gaps between those seconds emerge which might skew our functional connectivity. The brain is a fascinating and very complex part of our body; it is able to complete tasks in an extremely small amount of time. This means that in one second between the slices, the BOLD response could change dramatically. The procedure to correct this is to estimate the image of unknown slices based on known slices. For example, if at one second, the brain's activation is x, and at 2 seconds, it is $x + 10$, it is quite probable that the activation at 1.5 seconds is $x + 5$. By using this technique of estimation, we can account for missing values in our 4D fMRI volume. [Notter, 2016]

- Motion correction (also only used on fMRI) is used to correct for small head position changes during the course of an fMRI scan. [Notter, 2016] It is impossible for a human to keep their head perfectly still, especially over long periods of time, which creates a source of inconsistency. Since the head can rotate in all three dimensions (X axis, Y axis and Z axis), we need to account for movement in all three axes. To prevent this, a series of steps are taken. First, the mean head location in the X, Y and Z axes is computed by taking the average of the head position over the course of the fMRI's time series. Then, each slice is distorted so that the head position in it conforms to the mean head location. Thus, the head's location to be uniform over all the slices. [Notter, 2016]

- Artifact detection (only used on fMRI as well) is used to correct for large abnormalities in head movement. Rather than distorting an image that is significantly different compared to the average head location, artifact detection just removes it entirely. [Notter, 2016].

- The goal of coregistration is similar to artifact detection and motion correction—trying to keep head location uniform across slices. Coregistration maps the fMRI slice onto a predefined map. The map acts like tracing paper, and coregistration tries to move the slice around to get the perfect lineup. [Notter, 2016] Oftentimes,

an anatomical (sMRI) image will be used as the map for coregistration as it is at a fixed point in time. [Notter, 2016]

- Unlike other preprocessing procedures, normalization isn't used for correcting individual slices of an MRI volume. Rather, it is used for correcting entire volumes, so that the computer can compare "apples to apples" when trying to figure out the relationship between different scans.[Notter, 2016]. Normalization is used to correct for size and shape differences of different brains by mapping individual volumes onto a template space (also called an atlas- MSDL and Harvard Oxford are common). [Notter, 2016] This allows for a group comparison to be performed with brains of the same shape/size.

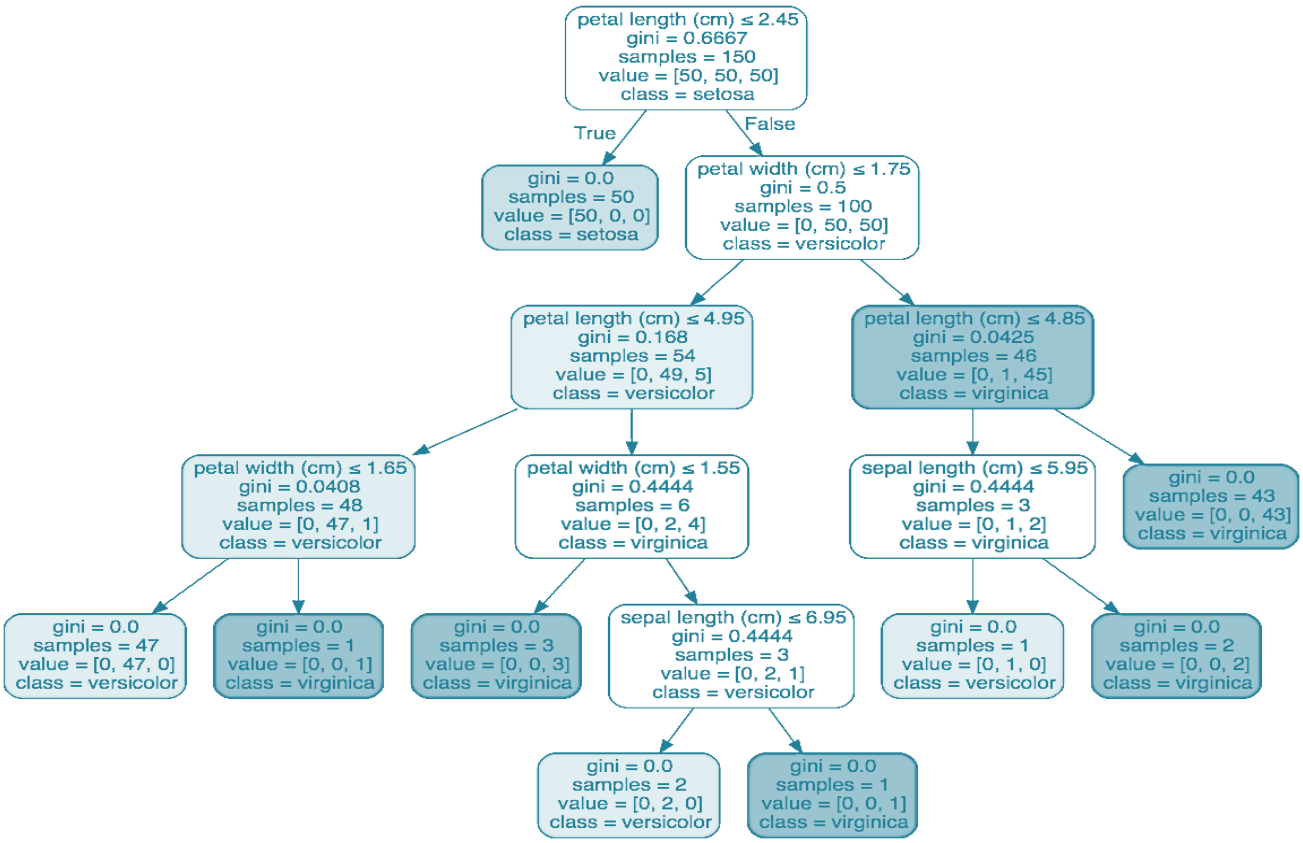
- Lastly, smoothing is used to reduce the smallest scale changes in the data, allowing for only bigger changes to be accounted for by our predictive model. [Notter, 2016]. This helps make larger scale changes more apparent to the computer- because the smaller ones are blurred. [Notter, 2016]. By applying a 3D Gaussian filter to the image (a bell curve like function, but in 3 dimensions), these smaller scale changes can become minimized and ignored by the computer. [Notter, 2016]. It is important, however, that the filter size is chosen carefully, otherwise different regions might get blurred together.

After preprocessing is done, the corrected scans can be fed into program which will supposedly return a diagnosis. The only problem with this is that computer programs need to be very specific-- every possibility needs to be programmed, otherwise an error will occur. With a program like a "choose your own adventure game", the rules are inherently very specific: if x is true, then do y. Otherwise, do z. However, when a very complicated statistical problem is presented, like recommending music based on previous song choices, it is impossible to break that down into the simple logic that computers require. This is where the field of machine learning, the backbone of artificial intelligence, comes in. Machine learning opens a whole new world of possibilities-- through a series of statistical models, computers can learn the relationship between different variables [Maini, 2017]. Suddenly, computers that could beat people in chess, control self-driving cars and give product recommendations. All of these tasks that were previously thought to be impossible, are now a reality because of the field of machine learning.

There are two basic schools of machine learning: supervised and unsupervised learning. Supervised learning is a technique in which the computer is given labeled data. Labeled data means that you have input data (also known as features) as well as the correct outcome (also known as labels). [Maini, 2017] In essence supervised learning looks at the input data and the correct output labels and via a machine learning algorithm, tries to recognize a pattern between

them. This is often called training or fitting the model. Using the relationship that the computer has found between the variables, a predictive model can be built. This allows a set of input data, without the correct outcome to be given to the trained model, and a predicted outcome will be returned. [Maini, 2017] However, sometimes you don't have labeled data, which is a common use case for unsupervised learning. One aim of unsupervised learning is to cluster similar portions of input data together [Maini, 2017]. Although unsupervised learning is a very intriguing technology, supervised learning will be the focus of this paper. Below is a very high-level description of different supervised learning algorithms-- also known as classifiers. Although this description should be sufficient for one to understand this experiment, the captivating mathematics behind each classifier is not explored, to keep the paper open to a wide audience.

- One of the simplest classification algorithms to conceptually understand is called the decision tree. A decision tree uses "if-then" statements to define splits in the data. [Yee, Chu, n.d.] However, instead of the programmer having to write in all of these if-then statements, the algorithm figures them out on its own. "For example, if a home's elevation is above some number, then the home is probably in San Francisco." (Yee, Chu, n.d.) "In machine learning, these statements are called forks, and they split the data into two or more branches based on some value." (Yee, Chu, n.d.) Like in any other supervised machine learning algorithm, input data without the correct output can be given into the trained classifier and a predicted outcome will be returned. For instance, if we give the computer a home elevation it can will predict whether the home is in San Francisco (given we provide it with enough examples). The decision tree figures out where to create these "if-then" statements by using a technique called Gini impurity. "Gini Index [impurity] is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower Gini index should be preferred. It helps us calculate how homogeneous the split in the data is" [Saxena, 2017]. By using this, the computer can create a simple, but powerful algorithm that can provide an effective implementation of machine learning.




[Fig A: A decision tree classifying different flowers from the popular Iris dataset] (Allison, 2016)

- Random Forests and Extra Random Forests are examples of ensemble classifiers [Benyamin, 2012]. Ensemble classifiers work by combining several "weak learners" in order to create a stronger one. For both of these ensemble classifiers, the "weak learners" are decision trees. This is not to say that decision trees are weak per se, but when you have more complicated data, random forests might be a better option. Random forests also help to reduce overfitting within the decision tree. Overfitting occurs when the classifier works well on the data it is trained on, but it doesn't work well on any other data; it is overfit (or over trained). [Benyamin, 2012]. Unfortunately, this is a very common problem with decision trees, which have a high probability of being overfit. [Benyamin, 2012]. Random Forest classifiers train decision trees on random subsets (hence the name random forests) of the original dataset and may duplicate or delete some data. Then, the voting majority wins and that majority is what the model outputs as its result. [Benyamin, 2012]. The Extra Random Trees classifier is identical to the Random Forests classifier except for one aspect. Unlike a random forest, at each step, the entire sample is used, and decision boundaries are picked at random, rather than

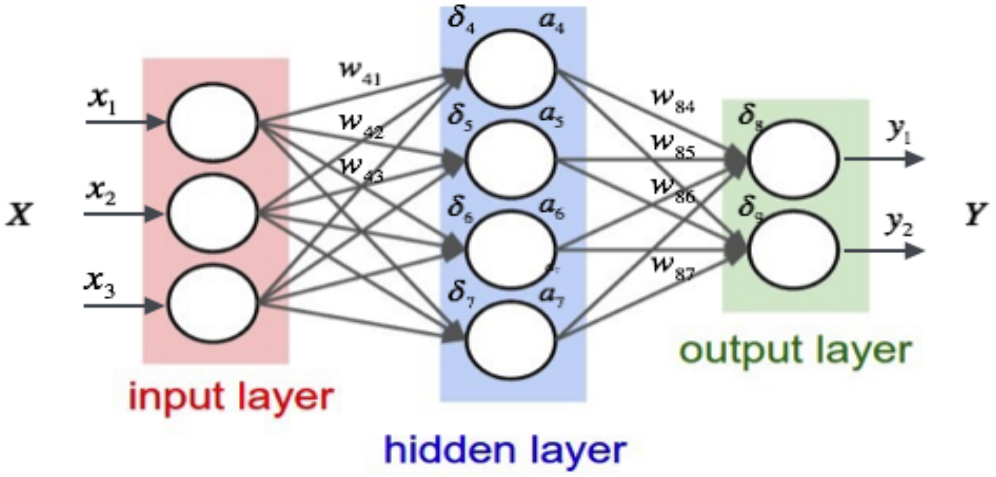
the best one. In some scenarios, Extra Random Forests Classifier outperforms Random Forests and the same thing is true in reverse. [Benyamin, 2012].

- Gradient Boosting is also an ensemble classifier and very similar to Random Forests. The difference between Random Forests and Gradient Boosting is that Random Forests trains on "fully grown" decision trees while Gradient Boosting only trains on child trees (that don't have a large depth). ["Gung" and "Antoine", 2015 and 2018 respectively.] A fully-grown decision tree is a decision tree that has as many splits as the algorithm deems appropriate while a tree stump is a decision tree with only one split. By combining several tree stumps and voting on the output, like in the Random Forests algorithm, the majority decision is returned. ["Gung" and "Antoine", 2015 and 2018 respectively.]

- The Bagging and the AdaBoost classifiers are also ensemble classifiers but are a lot more generic as the bagging and AdaBoost classifiers can be run on top of any other classifier. [Pedregosa, et. al, 2011] "A [The] Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate[s] their individual predictions (either by voting or by averaging) to form a final prediction." [Pedregosa, et. al, 2011] The bagging classifier is, in essence, a more generic version of the Random Forest classifier, except the final predictions have the option to be are averaged rather than voted on. Contrary to the Random Forests algorithm, however, the Bagging classifier can be run on top of any classifier, not just decision trees. "An AdaBoost [1] classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases." [Pedregosa, et. al, 2011] The AdaBoost classifier is very similar to the bagging classifier, except it weights classifiers that predict incorrectly less in the final average. [Pedregosa, et. al, 2011]

- The current "state of the art" classifier is called a neural network and is modeled after how the brain works. [Woodford, 2018]. In our brains, each neuron is composed of three components, the dendrite, the axon and the cell body. The dendrite is the cell's input; it is responsible for receiving information from other neurons. The axon is the cell's output; it is responsible for sending information via chemical transmissions to other neurons. Lastly, the cell body is where the "processing" of information happens. [Woodford, 2018]. Computer scientists sought out to create a machine learning algorithm modeled after how our brain

sends and receives information, and the result was the neural network. Rather than a series of chemical exchanges, neural networks are a series of mathematical operations.




[Fig B: A generic diagram of a "vanilla" Multi-Layer Perceptron classifier]

- Neural networks are structured in layers, and each layer is made up of neurons. In a traditional neural network, the output of each neuron in one layer influences the output of every neuron in the next layer (a fully connected NN). [Woodford, 2018]. However, during training, the influences are weighted, so some neurons might have more of an influence on others. Training is done through a process called backpropagation, which feeds the data backwards from the output to the input, and then adjusts the weights of connections to create the most accurate model. [Woodford, 2018] With all of the data fed into a neuron, the neuron activates or 'fires' based on an activation function (ReLU, linear, tanh, logistic, SoftMax etc.), and sends its output to every other neuron in the next layer. [Woodford, 2018] By using this process, computers are able to create very accurate models. A neural network, however, is a broad term that has a lot of subtypes including deep neural networks, multi-layer perceptrons, and deep belief networks. Keep in mind that these are just a few of the many different kinds of neural network algorithms.

- A deep neural network is simply a neural network with many layers. The term "deep" isn't a technical term, it is a marketing term. ["franck-dernoncourt", 2016]. This means that there aren't a specific number of layers a neural network needs in order to be considered "deep", but rather the term is used to talk about a multi layered neural network in general.

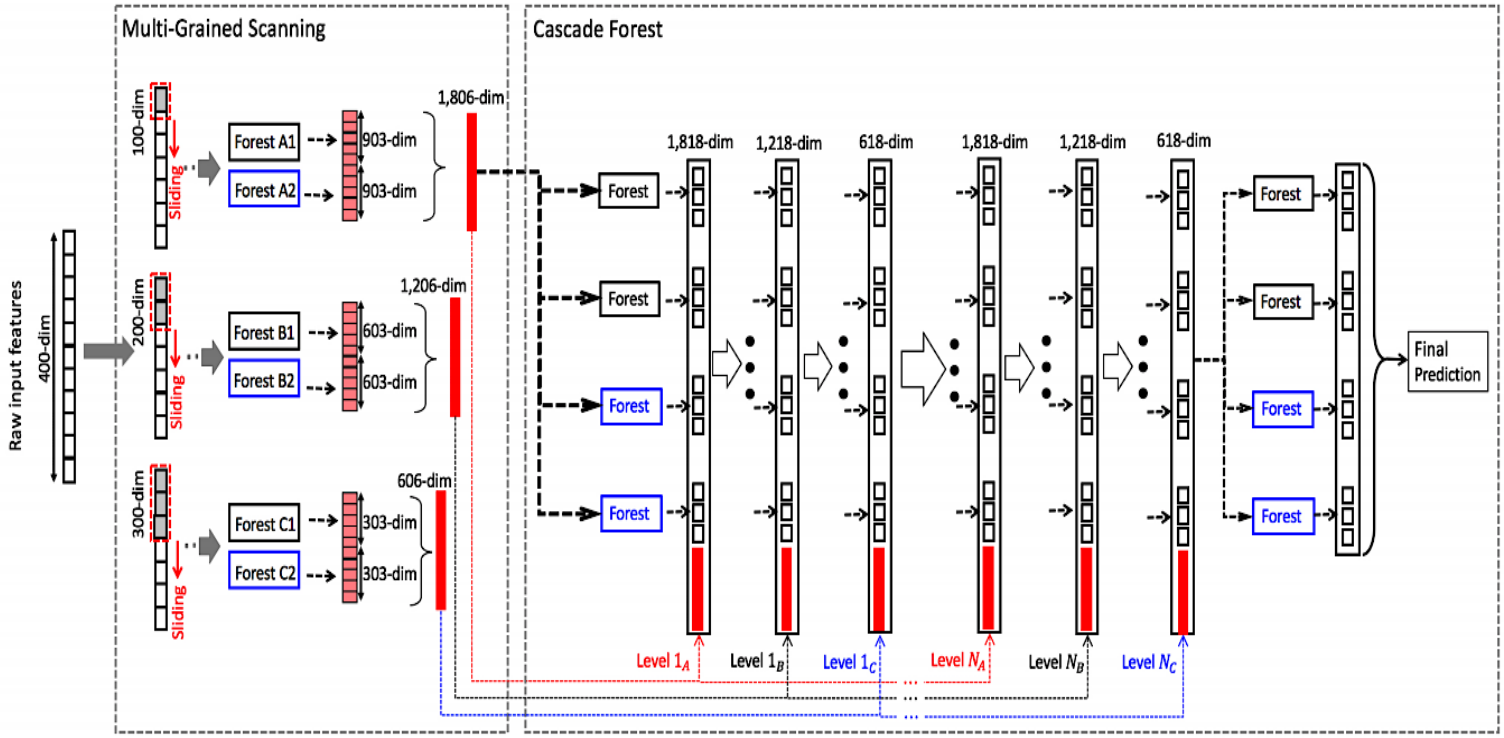
- A more specific kind of neural network is the Multi-layer perceptron. A multi-layer perceptron is a neural network that has at least three layers. It is a feedforward neural network meaning that all of the connections only move in a forward manner (layer 1 → layer 2 → layer 3). Also, all of the inputs are processed with a nonlinear activation function. [Multi-Layer Perceptron, 2018]

- A deep belief network is more complicated than a multi-layer perceptron or a deep neural network. Deep belief networks are special because they can be used with and without supervision (labeled data). [Deep Belief Network, 2018]. "When trained on a set of examples without labeled data, a deep belief network can learn to probabilistically reconstruct its inputs". [Deep Belief Network, 2018]. The layers then act as feature detectors (detecting important correlations in the data input) and can be further trained to perform classification. "A DBN can be looked at as a composition of Restricted Boltzmann Machines (RBMs) or autoencoders, which are both examples of unsupervised neural networks." [Deep Belief Network, 2018] "One important thing about DBNs that led to them becoming so powerful is the observation that they can be trained greedily, one layer at a time--creating a very effective deep learning algorithm". [Deep Belief Network, 2018]

- On a simpler note, the KNeighbors algorithm is a lot easier to understand. The KNeighbors estimator classifies points near each other on a graph (any dimensional space). The "K" in the KNeighbors classifier refers to the number of data point labels that the algorithm considers in making its decision. When you introduce a new point, the algorithm will measure the Euclidean distance to the "K" closest points ($distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ ) and vote on the label. For example, if the majority of the "k" closest points have the label '0', that is what the predictor will output.

- "Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome has to be a binary variable (either 0 or 1)." [Stanford UFLDL, n.d.] A logistic regression relies heavily on probabilities; the output that the model gives is a probability of the output being 0/1, given by a combination of the inputs. [Stanford UFLDL, n.d.] The probabilities are calculated by function called the logistic (or sigmoid) curve: $y = \frac{1}{1+e^{-x}}$. This equation gets closer and closer to 0 as x becomes smaller and closer and closer to 1 as it becomes bigger- making it perfect for probabilities. [Stanford UFLDL, n.d.] By using this curve, the logistic regression is able to give a very simple and elegant approach to classification.

- A Support Vector Classifier (SVC) is another type of machine learning classifier. [Wikipedia, 2018]. Like the KNeighbors classifier, the SVC classifier graphs the data in n-dimensional space (depending on the number of inputs/outputs in your data). [Support Vector Machine, 2018]. However, rather than classifying data based on the distance from other classes, the SVC tries to draw a boundary to divide the data points into 2 or more groups. [Wikipedia, 2018]. The SVC classifier has an input called a "kernel" which is how the boundary is drawn (linear, RBF etc.). [Support Vector Machine, 2018]. The SVC will use this kernel to draw a decision boundary to divide the data (if it is linear, it will draw a straight line to divide it and if it is nonlinear, it will draw some other dividing line). [Support Vector Machine, 2018]. Now if we introduce a new point, if it lies on one side of the boundary, we will classify it as one label, and if it is on the other, we will classify it as another label. [Support Vector Machine, 2018]. An SVC is




trained using an algorithm called gradient descent, which can be thought of iterating on the output incrementally until a spatial minimum is reached. [Support Vector Machine, 2018]. First, the computer draws a random line and measures the correctness of the boundary, and then it adjusts the line until the best one is found. [Wikipedia, 2018].

[Fig C: A full diagram of the gcForest algorithm] [Zhou, Zhi-Hua & Feng, Ji., 2017]

- The last algorithm we will discuss is relatively new- the Multi-Grained Cascade Forest (gcForest). gcForest was proposed in early 2017 as an alternative to deep neural networks. [Zhou, Zhi-Hua & Feng, Ji., 2017]. Their argument was that deep neural networks have too many input parameters that need manual adjusting, and that the training time and processing power needed to run them is very large. "[gcForest] It is a decision tree ensemble approach with performance highly competitive to deep neural networks in a broad range of tasks. In contrast to deep neural networks which require great effort in hyper-parameter tuning, gcForest is much easier to train; even when it is applied to different data across different domains in our experiments, excellent performance can be achieved by almost same settings of hyper-parameters." [Zhou, Zhi-Hua & Feng, Ji., 2017]. The gcForest is based off of the cascade structure of a neural network, but it operates quite differently. [Zhou, Zhi-Hua & Feng, Ji., 2017]. Each layer is made up of a couple of classifiers, rather than nodes. Inputs are fed into the first layer, where each classifier predicts and then the outcomes are voted on, creating a vector of the predictions. [Zhou, Zhi-Hua & Feng, Ji., 2017]. The vector of predictions, along with the initial vector of input data is then passed onto the next layer which predicts, votes and passes data onto the next, until it reaches a final prediction. [Zhou, Zhi-Hua & Feng, Ji., 2017]. To prevent overfitting, the model does cross validation on each layer, to make sure that predictions are valid. The gcForest algorithm also employs a technique they called Multi Grained Scanning, which allows for "representation learning". [Zhou, Zhi-Hua & Feng, Ji., 2017]. This allows for not only relationships between features and outputs to be found, but also relationships between the features themselves. [Zhou, Zhi-Hua & Feng, Ji., 2017]. Even though there has been some speculation on the validity of the testing in their paper, as well as the robustness of the algorithm itself, I felt that this algorithm could be worth trying and testing to see if it could be a robust classifier for my prediction task.

For this project, I chose to use the ADHD200 dataset which contains MRI scans from 362 ADHD subjects and 585 typically developing subjects (control group). The patients whose scans were taken were 7-21 years old and personally identifiable information was removed before release. [Pierre Bellec, et. al, 2017]. In the dataset, each subject has one or more resting state fMRI (no stimulus), one or more anatomical (sMRI) scans and accompanying phenotypic information (gender, IQ, age, handedness etc.). In the summer of 2011, the ADHD200 consortium challenged teams to create the best model they could using their dataset. [Pierre Bellec, et. al, 2017]. The consortium was created in the hope that if the consortium could distribute a dataset to the public, they could find a better solution for diagnosing ADHD. [Pierre Bellec, et. al, 2017]. The data in the ADHD200 dataset came from 7 sources: Brown University (although this data was excluded in the actual competition dataset, so it doesn't count), Peking University, New York University Child Study Center, Oregon Health and Safety university,

Washington University in St. Louis, University of Pittsburgh, NeuroIMAGE, and the Kennedy Krieger Institute. [Pierre Bellec, et. al, 2017]. In sharing this data, the consortium realized the importance of reaching beyond the imaging community, which typically consists of psychiatrists, neurologists, and neuroscientists, to broader multidisciplinary scientific disciplines. [Pierre Bellec, et. al, 2017]. "To recruit the global scientific community to address childhood psychiatric illness, a competition was announced, with the goals of developing: (1) novel strategies for predicting diagnostic status based on an individual's intrinsic functional architecture and brain structure, and (2) novel techniques for identifying brain features that may yield ADHD biomarkers." [Pierre Bellec, et. al, 2017]. However, in the competition, they weren't only trying to diagnose ADHD, but also the subtypes of it (ADHD-Hyperactive, ADHD-Inattentive, ADHD-Combined [both hyperactive and inattentive]). In this paper, I chose to only try to discern ADHD from non-ADHD patients.

The aim of this project is not to create a clinical replacement for the diagnosis of ADHD through MRI scans and patient data. Rather, it is to improve on current known model performance and create the basis for future papers on this subject, so eventually, an algorithm can become the "gold standard" for ADHD diagnosis. As I said above, ADHD overdiagnosis is a huge issue that plagues our society. As someone who has ADHD, it is frustrating that a lot of times, people tell me that ADHD isn't a "real" disorder-- it is just something made up by lazy parents or used by lazy kids as an excuse to slack off. I hope that through this project, I can make a difference and show kids that ADHD is a real disorder, and not an excuse.

My initial hypothesis when starting this project was that using newer, state of the art models like the Random Forests algorithm or Gradient Boosting (using the xgboost library) was going to give the highest accuracy. I based this off the fact that ensemble classifiers (models that use multiple versions of other models) usually work well on small datasets like ours. Additionally, the runtime of Random Forests and Gradient Boosting is very fast, so I knew I could run a lot of optimizations without a massive and expensive computing resources. My aim was to improve on performance above current known prediction levels from the ADHD200 competition.

# Procedures

## Materials

As discussed in the introduction, the dataset that I used was called the ADHD 200 dataset. It contains 362 ADHD subjects and 585 typically developing subjects (or non-ADHD subjects). I was able to download the dataset, preprocessed using the Athena pipeline (a preprocessing pipeline created by the ADHD200 consortium) upon approval by an administrator from the NITRC website. The Athena preprocessing pipeline was used to smooth, normalize, mask and map the data to enable standardization across 8 different locations at different times and under slightly different conditions. This pipeline is the most robust, widely used, and best documented of all alternative preprocessing pipelines applied to the ADHD200 dataset. There were 8 phenotypic files inside the data from each site-- the first one was a "global" phenotypic file containing the labels (ADHD-Combined, ADHD-Inattentive, ADHD-Hyperactive or Typically Developing), and the other 7 (site specific) contained more specific data (patient age, gender, dexterity, medication status, IQ etc.).

The primary neuroimaging library I used for this project is called *nilearn* and is implemented in Python. It provides a fast and flexible API (application programming interface - a set of rules and methods that one can integrate into their program) for manipulating NiFTi files (the most common MRI format). Nilearn already had a method (a function) to retrieve a subsample from the ADHD 200 data set, but it only retrieved 40 images, which I didn't feel would be utilizing the dataset to its full potential. To fix this, I downloaded the data onto a hard drive, and wrote a script to find the paths of each resting state fMRI scan and created an object from it, that was similar the 40 images object that nilearn provided (attached in the appendix and also on GitHub). The script looked into the "global" phenotypic file and found the label for each subject, so that we could do supervised machine learning.

## Iteration 0 - Keras CNN using raw fMRI volumes

### Objective

The objective in the first iteration was to test the raw data on a simple neural network model to see if it would demonstrate any signal.

### Procedures

Without prior knowledge of the Athena pipeline, I used a library called nipype to do preprocessing on the data (normalization, slice time correction etc.) and then fed it into 4D CNN (convolutional neural network - a neural network that does matrix convolutions, or matrix multiplications on images to find features, because an image is essentially a matrix of RGB values for each pixel). I built this CNN using a library called Keras, which makes it easy to build neural networks using Python.

**Results**

However, this didn't give me a good model- it predicted all non-ADHD every time. Additionally, it took an extremely long time to train (around 3 days, even on GPUs (graphics processors)), and also used a lot of computing capacity, because each scan is around 200MB (roughly equivalent to a 10-minute low resolution video). Even though using a CNN might have contributed to the flawed model, I felt that the sheer run time and computing requirements made feeding the basic preprocessed data into a machine learning model impossible.

**Revisions**

I reached out to Dr. Kai Zhang, Ph.D., a Post-Doctoral Research Fellow at the Stanford University School of Medicine, because I needed some input on how to proceed. He recommended that I use "functional connectomes", a method used in analysis of brain images to reduce the amount of data, while still keeping useful information in the scans.

"Functional connectomes capture brain interactions via synchronized fluctuations in the functional magnetic resonance imaging signal. If measured during rest, they map the intrinsic functional architecture of the brain. [...] Analyzing their variability across subjects and conditions can reveal markers of brain pathologies and mechanisms underlying cognitivity" (Varaquaux & Craddock, 2013). In essence, it measures the variance between MRI signal response in different regions of the brain. Functional connectivity can be a very effective tool to gathering useful information from a large amount of data. To extract these connectomes, we have to specify something called a "mask", or the set of regions for the computer to look at when analyzing the connectivity. These set of regions are found is something called an *atlas*. Luckily, nilearn, a Python library for neuroimaging analysis, provides an easy to use API to make this process easy. We can apply this to each NiFTi volume and retrieve a signal from the volume. The *Nilearn* API provides five different functions to extract these connectivity signals: correlation, partial correlation, tangent, precision and covariance. I decided to focus on tangent, as, based on nilearn documentation, it is the most cutting-edge technique of feature extraction because it includes both correlation and partial correlation in its calculation.

# Iteration 1 - Common ML Algorithms with functional connectivity using the MSDL atlas

## Objective

The objective of this iteration was to use functional connectivity as a feature in an initial exploration to get a sense of which models were worth focusing on in the future, using a simple pipeline, brain atlas, and machine learning library.

## Procedures

I rewrote my Python script to extract functional connectomes from each NiFTi scan using the MSDL atlas (a common atlas for mapping brain regions) with *nilearn.* By doing this, I was able to turn the verbose and complex four-dimensional raw fMRI data into a one-dimensional vector, which is a lot easier for the computer to process. In all, we now had 417 features that the model could use for its prediction. I tried using a lot of the most common models trained on the ADHD200 supplied training and testing datasets and had varying degrees of success.

## Results

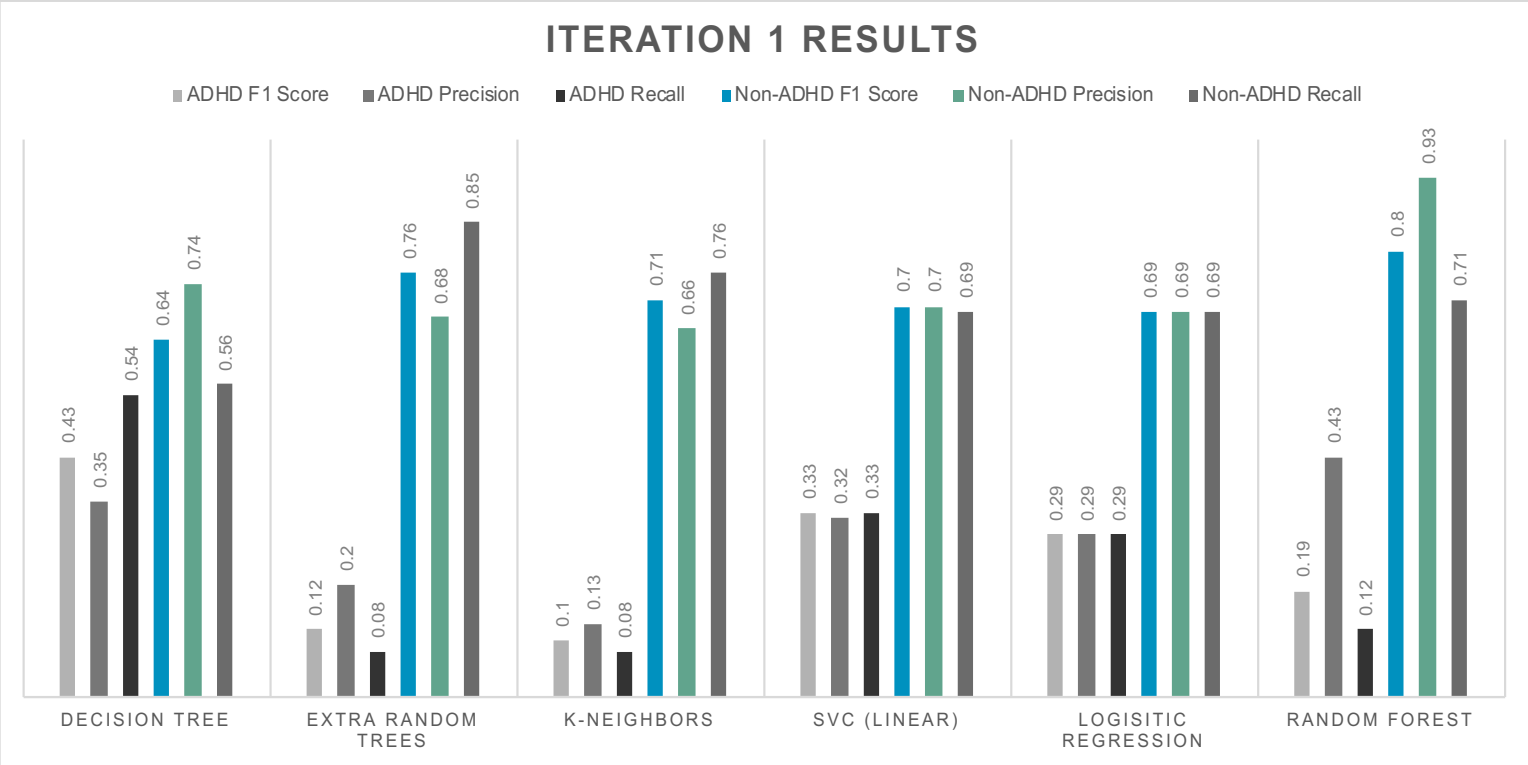
To evaluate my model, I used the most commonly used metrics to evaluate classification models: F1 score, Precision and Recall. To understand these, a common example used is a test that you might take in school. If you take a test and answer 7 out of 10 questions, but within those 7 questions you answered you answered 5 correctly, your recall is 70%, or 7/10 (how many you answered) and your precision is 5/7 or 62.5% (how many you answered correctly within the ones you answered). To evaluate both precision and recall performance, the F1 score was created as another metric to measure the correctness of a model. F1 score is the harmonic mean of precision and recall. It is calculated with $F1 = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$. Using this metric, we can get a good idea of how accurate a model is, when we calculate the precision, recall and F1 score for both ADHD and Non-ADHD labels. Another metric that was explored during this project, was accuracy. However, accuracy ($\frac{correct\ classifications}{total\ classifications}$) didn't seem like a significant metric as a model could get a 66% accuracy by predicting all non-ADHD, because ⅔ of the dataset is non-ADHD patients.

The model was tested on the ADHD200 supplied training and testing datasets using these classifiers: Decision Tree, Random Forests, Extra Random Forests, K Nearest Neighbors, SVC with a linear kernel, and Logistic Regression. The connectomes were extracted through the tangent method.

| Metric | Decision Tree | Extra Random Trees | K-Neighbors | SVC (Linear) |
|---|---|---|---|---|
| ADHD F1 Score | **0.43** | 0.12 | 0.1 | 0.33 |
| ADHD Precision | 0.35 | 0.2 | 0.13 | 0.32 |
| ADHD Recall | **0.54** | 0.08 | 0.08 | 0.33 |
| | | | | |
| Non-ADHD F1 Score | 0.64 | 0.76 | 0.71 | 0.7 |
| Non-ADHD Precision | 0.74 | 0.68 | 0.66 | 0.7 |
| Non-ADHD Recall | 0.56 | **0.55** | 0.76 | 0.69 |

| Metric | Logistic Regression | Random Forest |
|---|---|---|
| ADHD F1 Score | 0.29 | 0.19 |
| ADHD Precision | 0.29 | **0.43** |
| ADHD Recall | 0.29 | 0.12 |
| | | |
| Non-ADHD F1 Score | 0.69 | **0.8** |
| Non-ADHD Precision | 0.69 | **0.93** |
| Non-ADHD Recall | 0.69 | 0.71 |

[Table 1. Positive and Negative F1, Precision and Recall from the first iteration of this experiment]




[Fig D: Graph of Metrics from Iteration 1]

As we can see from *Table 1* and *Fig C,* functional connectivity, even on the default implementations of each machine learning algorithm, provides an excellent baseline for future iterations. It seems, from *Table 1 and Fig C,* especially that the decision tree based models seem to have the best performance across the classifiers we tested. Decision tree based models include, but are not limited to decision trees, extra random forests and random forests. Not only do they seem to provide high averages of results, but they also have certain strong points that make them a good choice to proceed with. For example, Random Forests has a recall of 0.93, meaning that out of the patients it classified as not having ADHD, 93% of them didn't have the disease. Additionally, Extra Random Forests was able to recognize 85% of non ADHD patients. Lastly, the decision tree provided to be a good average model, with an ADHD F1 score of 0.43, it is higher than any of the other models we tested in this iteration.

However, it is interesting to note that all of the non-decision tree classifiers seem to be very stratified between ADHD and non ADHD metrics. Logistic regression, for example, has an ADHD F1, Precision and Recall of 0.29 and a non ADHD F1, precision and recall of 0.69. Similarly, the SVC with a linear kernel predicted 0.33, 0.32 and 0.33 for ADHD metrics, respectively and 0.7, 0.7 and 0.69 respectively for non ADHD. For K-Nearest Neighbors, the stratification isn't as clear as the results within each class (positive ADHD or negative ADHD)

vary by around 0.08, but the same division is still present. One plausible explanation for this strange result could be that the non-decision tree classifiers were predicting completely random results; 0.7 and 0.3 are roughly the division between non ADHD and ADHD samples in the training/test set.

**Revisions**

After this iteration, I realized that changes needed to be made. First, I realized that we needed a method to prevent overfitting when evaluating a classifier. As a result, I decided to implement a new method of evaluation. Instead of using the ADHD200 supplied training and testing datasets, I combined them into one large dataset. Using the *train_test_split* method in scikit-learn (an excellent Python machine learning library), I used a random 20% as a test dataset and the remaining 80% for training. Using a specified number of iterations to use for cross validation, I revised the script to run the model the specified cross validation times, using a different 20% each time, and average the F1, Precision and Recall over each iteration. By doing this cross validation, we can make sure that overfitting isn't an issue.

Second, after reading a couple of papers, in which advanced neural networks were used to achieve high accuracies, I decided to also explore using Multi-Layer Perceptron Classifiers and Deep Belief Networks.

Third, I decided that one way we could improve the model's accuracy was by adding another feature. The natural feature to incorporate seemed to be phenotypic information. Unfortunately, a lot of the fields inside the site-specific phenotypic files were missing values, which can degrade the performance of a classifier. However, there were columns that were (mostly) filled out: age, dexterity (handedness) and gender. To include these, I decided to write a separate model using only phenotypic information, which was a Random Forest with a Bagging Classifier. In the next iteration of models, I included the probability that a given subject was ADHD solely from the phenotypic information as a feature. In order to use this data, without having it conflict with the functional connectivity, I surrounded the probability with 0s with the hope that if every subject's probability was surrounded by 0s, it wouldn't be picked up as an important feature

Fourth, I decided to switch the atlas from MSDL to HarvardOxford as it seemed logical to experiment with different brain maps. I thought that it might have a positive impact on the usefulness of the functional connectivity.

Lastly, in order to make the data easier to understand for the neural networks, I decided to equalize the data- by removing non-ADHD data so that ADHD samples and non-ADHD samples have the same number of samples.

# Iteration 2 - Multi Layer Perceptrons and Deep Belief Networks on Functional Connectivity and Phenotypic Information using Cross Validated Optimization

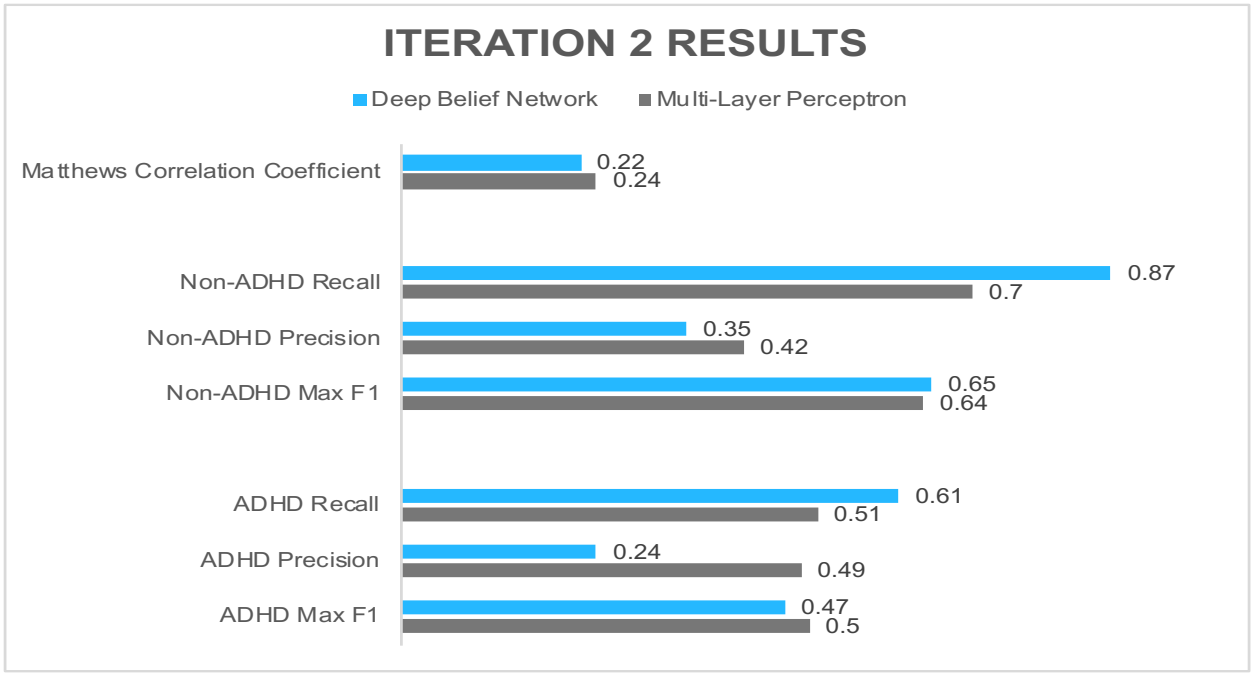
### Objective

The objective of this iteration was to apply newer neural network models in a rigorous way to develop a baseline understanding of how well they could perform on the ADHD200 dataset.

### Procedures

Rather than using a scikit-learn algorithm like GridSearchCV or RandomizedSearchCV, to find optimal hyperparameters, because they are computationally expensive as well as only rank by one accuracy metric, I used the optimizer that I specified in the last revisions section. With the MLP and DBN, after lots of experimentation, I found that the hyperparameter that most dramatically affected the F1, precision and recall was the number of nodes in a layer. Unfortunately, there is no real rule of thumb to determine the optimal number of layers and nodes, besides experimentation. By using the optimizer I wrote to try all node values up to 500 in a one layer DBN/MLP (to make it simpler and take less time to run) with 4 cross validation iterations, I was able to locate a couple of layer sizes that stood out. The ADHD probability from the phenotypic model (RF + Bagging) was included as a feature with this model, along with the functional connectivity

### Results




[Figure E: Metrics from Iteration 2]

| Metric | Multi-Layer Perceptron (fMRI) + Random Forest (w bagging) for phenotypic | Deep Belief Network (fMRI) + Random Forest (w bagging) for phenotypic |
|---|---|---|
| | | |
| ***Optimal number of layers*** | *403* | *434* |
| | | |
| **ADHD F1** | **0.5** | 0.47 |
| **ADHD Precision** | 0.49 | 0.74 |
| **ADHD Recall** | **0.51** | **0.19** |
| | | |
| **Non-ADHD F1** | 0.64 | **0.65** |
| **Non-ADHD Precision** | 0.42 | 0.35 |
| **Non-ADHD Recall** | 0.7 | **0.87** |
| | | |
| **Matthews Correlation Coefficient** | **0.24** | 0.22 |

[Table 2: *Optimal Layer Numbers*, Positive and Negative F1, Precision, Recall and MCC.]

The results from *Table 2* and *Figure B* lead us to believe that in fact Multi-Layer Perceptron and Deep Belief Networks, even with coupled cross validation, perform better than the first iteration. The cross validation is important to note as testing our algorithm on five different random subsets of data is a **much** harder test than the first iteration. Regardless our ADHD F1 was higher in both iterations, which signifies that this might be a better model. Since the decision tree proved to be the best average model in the previous iteration, we can use it as a comparison point for this iteration. 0.43, 0.64. The ADHD F1 score for the Decision Tree in the first iteration was 0.43, and in this iteration, for DBN, it was 0.47. Although a change of 0.04 might not seem like a lot, like I said before, it is important to note the harshness of our test. For the Multi-Layer Perceptron, we were able to achieve an even better ADHD F1 score of 0.5-- 0.07 better than the decision tree. For non ADHD F1, the F1 scores didn't differ drastically-- the decision tree non ADHD F1 score was 0.64 while the MLP also had 0.64 and DBN, 0.65. This large change in ADHD F1 scores shows that not only do neural networks provide a more effective and accurate alternative to diagnosing ADHD, but also a much more robust one.

However, neural networks are not without their weak points-- the DBN was only able to recognize 19% of ADHD patients which definitely is sub optimal. For the Multi-Layer Perceptron, within the subjects it diagnosed as having ADHD, only 47% of them actually had the disease. Still, these results, especially using this harsher test, shows that using neural networks on functional connectivity and random forests + bagging on phenotypic information provides an excellent tool for ADHD diagnosis.

In this iteration, we introduced Matthews Correlation Coefficient which I found to be a very telling metric as to how well a certain model performs. The Matthews Correlation Coefficient is calculated, like a lot of other metrics, from a confusion matrix. A confusion matrix is a simple table that allows you to see the results from a classifier.

| | **Predicted: NO** | **Predicted: YES** |
|---|---|---|
| **Actual: NO** | True Positives | False Positive |
| **Actual: YES** | False Negative | True Negative |

[Fig F: Diagram of a confusion matrix]

Matthews Correlation Coefficient is calculated with this equation, where TP = True Positives, FP = False Positives, FN = False Negatives and TN=True Negatives: $MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}}$. The reason that MCC is such a powerful metric for our scenario is that if the result is random (there is no trend between actual and predicted, like it predicts all 1s or 0s), it gives an MCC of 0.0. If the predicted results are completely the opposite of the actual, the MCC is -1.0, while if they are exactly the same, our MCC is 1.0. This allows us

to see how our result compares to a completely random result. The fact that these neural networks are able to achieve an MCC of 0.24/0.22, shows that there is in fact a trend between the inputs and the outputs. This is a very motivating result.

**Revisions**

Although employing neural networks was a step in the right direction, they took too long to run, as well as require tens of thousands of samples to work effectively, rather than hundreds like our dataset. After reading the gcForest paper, which was proposed as an alternative to neural networks-- requiring much less data and processing power, I decided to implement it

I decided to make three major changes for the next iteration. The first one regarded phenotypic information. Instead of using the phenotypic probability as a feature into the gcForest model, I used the phenotypic information (gender, dexterity and age) directly as features into the gcForest model. The gcForest algorithm was providing a much higher MCC than the phenotypic model, so I decided that it was worthwhile to try putting a new feature into the better performing model.

Second, I decided to write an automated optimizer for the gcForest algorithm. Randomly, the optimizer altered gcForest parameters such as the order of the layers, the classifiers used and the parameters of those classifiers. It exports the mean, max, min, std recall, precision, f1, accuracy and MCC into a CSV, so tools like Apache Spark (a tool to gather insights from data by writing MapReduce type scripts), SQL (a database language) and simple spreadsheet applications like Excel can be utilized to gather insights from these massive files.

Lastly, I extended the default gcForest classifier to use two new classifiers, LinearSVC and Multi-layer Perceptron. In previous iterations, those two results have been very promising, so using them with gcForest's cascade structure seemed like a good idea. By adding these classifiers as possibilities for the optimizer to pick, I hope to achieve a better model.

# Iteration 3- Computerized Random Search on Phenotypic Information (as a feature) and fMRI volumes using gcForest

### Objective

My objectives in this last iteration were to improve my integration of phenotypic features in the model, to use an automated optimization technique and most of all, to use the gcForest model to see if I could improve on the performance of neural network models from Iteration 2.

### Procedures

For this iteration, as I mentioned in the previous iteration's *revisions* sections, I used a computerized random searching to find optimal parameters for the gcForest algorithm. This is necessary as not only do the gcForest hyperparameters need to be tuned, but each individual classifier inside gcForest's cascade structure also needs to be tuned. Here are the parameters:

I considered the following classifiers:

1) Multi-Layer Perceptron Classifier (MLP)
2) Support Vector Machine Classifier (SVC)
3) Logistic Regression
4) XGBoost Gradient Boosting Classifier (XGB)
5) Random Forests (RF)

The computerized searcher altered these attributes:

1) Number of Logistic Regressions between each classifier chosen
2) MLP Solver (adam, lbfgs, sgd)
3) SVC Kernel (polynomial, radial basis, linear)
4) Number of XGB Estimators (up to 130 by default)
5) Number of Random Forest Estimators (up to 130 by default)
6) The order of the models (xgb-->logit or svc-->mlp)
7) Early stopping iterations (how many iterations in a row need to be below the best average for the model's training to be terminated)
8) Whether to use each classifier (varied probabilities)

As I mentioned previously cross validated results running over 5 random subsets of the data were exported to a CSV file.

We considered mostly considered the mean, maximum, minimum and standard deviation (STD) of these metrics

1) F1 Score (mean, min, max, std)
2) Accuracy (mean, min, max, std)
3) Precision (mean, min, max, std)
4) Recall (mean, min, max, std)
5) Matthews Correlation Coefficient (mean)

I ran this searcher on two Amazon EC2 instances (*c5.xlarge*) as well as various old laptops from around the house. By aggregating all of the CSV files and running Apache Spark, and loading it into Microsoft Excel, I was able to find a couple of parameter combinations that stood out.

**Results**

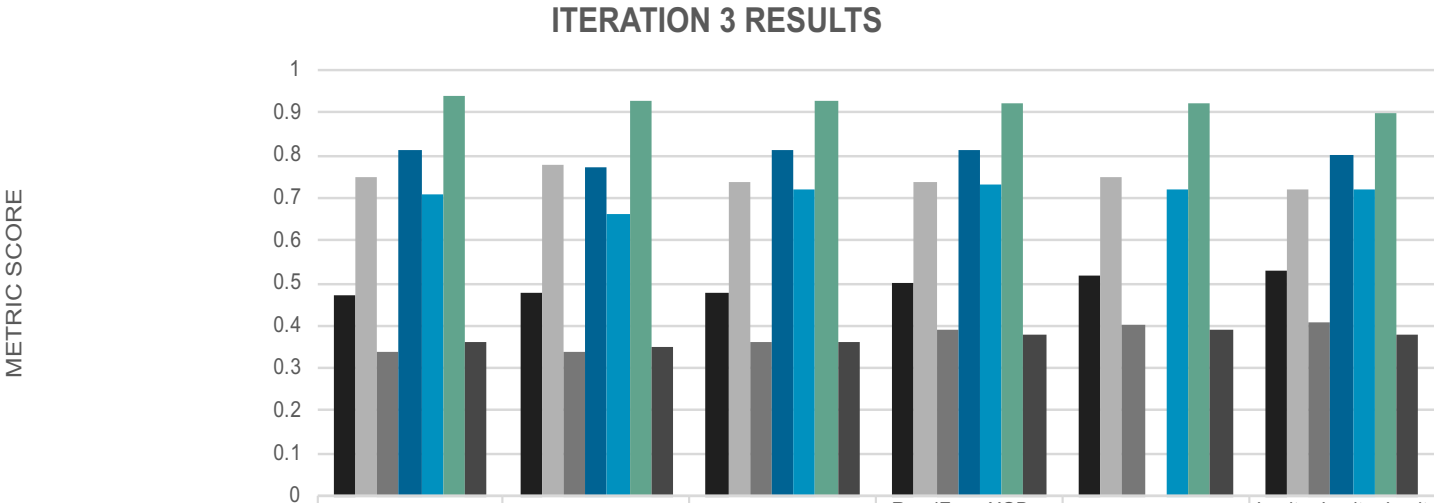
| Metric | GC Forest: Optimal 1 | GC Forest: Optimal 2 | GC Forest: Optimal 3 |
|---|---|---|---|
| | | | |
| ***Layer Order*** | *SVC > Logit3* | *RandFor > Logit > RandFor* | *Logit > MLP > XGB > MLP* |
| ***MLP Nodes*** | | | *(114); (11)* |
| ***MLP Solver*** | | | *SGD, SGD* |
| ***SVC Kernel*** | *Poly* | | |
| ***XGB Estimators*** | | | *(76)* |
| ***Random Forest Estimators*** | | *(34); (134)* | |
| ***Early Stopping Iterations*** | *2* | *4* | *1* |
| | | | |
| **ADHD F1** | 0.47 | 0.48 | 0.48 |
| **ADHD Precision** | 0.75 | | 0.74 |
| **ADHD Recall** | 0.34 | 0.34 | 0.36 |
| | | | |
| **Non-ADHD F1** | 0.81 | 0.77 | **0.81** |
| **Non-ADHD Precision** | 0.71 | 0.66 | 0.72 |
| **Non-ADHD Recall** | **0.94** | 0.93 | 0.93 |
| | | | |
| **Matthews Correlation Coefficient** | 0.36 | 0.35 | 0.36 |

[Table 3: Results Part 1 of 2- models 1 to 3 from Iteration 3]

| Metric | GC Forest: Optimal 4 | GC Forest: Optimal 5 | GC Forest: Optimal 6 |
|---|---|---|---|
| | | | |

| | | | |
|---|---|---|---|
| ***Layer Order*** | *RandFor > XGB > XGB > Logit > RandFor* | *MLP > MLP > MLP > Logit* | *Logit > Logit > Logit > RandFor > RandFor* |
| ***MLP Nodes*** | | *(93,114); (136,29); (21,75,132)* | |
| ***MLP Solver*** | | *LBFGS; SGD; SGD* | |
| ***SVC Kernel*** | | | |
| ***XGB Estimators*** | *(57); (3)* | | |
| ***Random Forest Estimators*** | *(106,108)* | | *(15);(77)* |
| ***Early Stopping Iterations*** | *3* | *2* | *3* |
| | | | |
| **ADHD F1** | 0.5 | 0.52 | **0.53** |
| **ADHD Precision** | 0.74 | 0.75 | 0.72 |
| **ADHD Recall** | 0.39 | 0.4 | **0.41** |
| | | | |
| **Non-ADHD F1** | **0.81** | **0.81** | 0.8 |
| **Non-ADHD Precision** | **0.73** | 0.72 | 0.72 |
| **Non-ADHD Recall** | 0.92 | 0.93 | 0.9 |
| | | | |
| **Matthews Correlation Coefficient** | 0.38 | **0.39** | 0.38 |

[Table 4: Results Part 2 of 2 - models 3 to 6 from Iteration 3]




| | SVC > Logit3 | RandFor > Logit > RandFor | Logit > MLP > XGB > MLP | RandFor > XGB > XGB > Logit > RandFor | MLP > MLP > MLP > Logit | Logit > Logit > Logit > RandFor > RandFor |
|---|---|---|---|---|---|---|
| ADHD F1 | 0.47 | 0.48 | 0.48 | 0.5 | 0.52 | 0.53 |
| ADHD Precision | 0.75 | 0.78 | 0.74 | 0.74 | 0.75 | 0.72 |
| ADHD Recall | 0.34 | 0.34 | 0.36 | 0.39 | 0.4 | 0.41 |
| Non-ADHD Max F1 | 0.81 | 0.77 | 0.81 | 0.81 | 0 | 0.8 |
| Non-ADHD Precision | 0.71 | 0.66 | 0.72 | 0.73 | 0.72 | 0.72 |
| Non-ADHD Recall | 0.94 | 0.93 | 0.93 | 0.92 | 0.92 | 0.9 |
| Matthews Correlation Coefficient | 0.36 | 0.35 | 0.36 | 0.38 | 0.39 | 0.38 |

[Fig G: Results from each layer order combination]

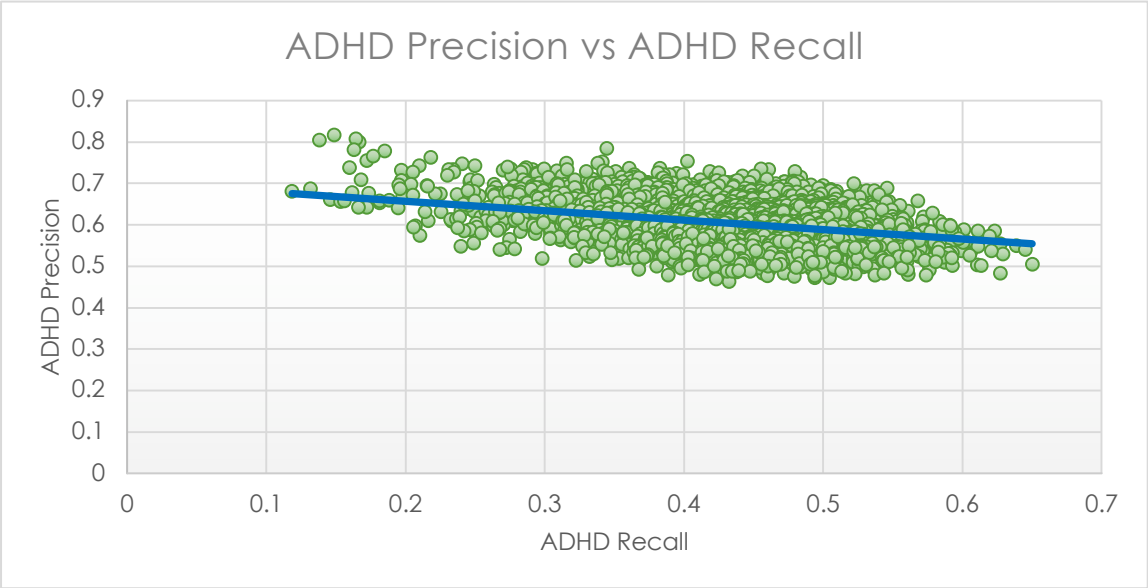
As evident from the tables and graphs above, gcForest provides a much more accurate and robust solution to diagnosing ADHD than any of the other classifiers we experimented with. The Non ADHD Recall doesn't go below 0.9 each entry in *Table 3,* which is much improved from previous iterations. Additionally, the Matthews Correlation Coefficient increased by almost 0.1 from the neural network, reinforcing the fact that there is a reasonably strong trend between functional connectivity, phenotypic information (as a feature) and the labels assigned to each subject. After analysis through Apache Spark and Microsoft Excel, I concluded that gcForest optimizer number 5 has the best average result-- which is furthered by the 0.39 MCC. However, as you can see from the tables, each model has their own unique strong point and also some common weak points.

Within the models included in *Table 3,* the maximum ADHD recall attained is 0.4 (gcForest optimizer #5). This shows a definite area for improvement. With the Multi-Layer Perceptron in iteration 2, a mean ADHD recall of 0.51 was achieved, showing that gcForest, although it has very promising results, struggles to recognize ADHD patients. However, within the patients it did recognize, 75% of them actually had the disease.

In iteration 1, we found that decision tree based classifiers gave the best results. However, a lot of the models in the best gcForest cascade structures had other classifiers, such as SVC and Logistic Regression. This is interesting because as we speculated, this could suggest a

random prediction from these models. When used in the gcForest, they seemed to make more powerful models, rather than random ones.

Also, within the table, there is a noticeable trend that as precision goes up, recall goes down and vice versa. As an experiment, I graphed the ADHD Precision vs ADHD Recall to see if there was in fact a trend.




[Fig H: ADHD Precision vs ADHD Recall (using all results: ≈10,000 combinations)]

As we can see from the graph above, there does seem to be a negative correlation between precision and recall, which was interesting to find out. It is also noteworthy that the MCC for this iteration is higher than previous ones, where the best average result (#5) almost has a 0.4 MCC vs a baseline which would be 0. Additionally, the negative mean F1 is 0.81, showing that we have a very powerful model for recognizing non-ADHD patients. This shows that gcForest provides a very robust, accurate and powerful model for diagnosing ADHD from functional connectivity and phenotypic features.

# Discussion

**Sources and issues in results improvement over study iterations**

| | Iteration 1 | Iteration 2 | Iteration 3 |
|---|---|---|---|
| **Non ADHD F1** | **0.64** | **0.64** | **0.81** |
| **ADHD F1** | **0.43** | **0.5** | **0.52** |
| **MCC** | | **0.24** | **0.39** |
| **What Changed?** | MSDL Atlas<br>No Phenotype Info<br>Untuned (default) sklearn classifiers<br>Athena Pipeline<br>No CV<br>Best (shown in table): Decision Tree | Harvard-Oxford Atlas<br>Phenotypic probability as feature<br>Neural network models<br>5 iteration CV<br>Hidden layer searching- best**:** single layer: 403 nodes<br>Equalized Data<br>Best: MLP | Harvard-Oxford Atlas<br>Phenotypic information, surrounded by 0s as feature<br>gcForest models<br>5 Iteration CV<br>Computerized random searching<br>Unequalized data<br>Best: gcForest-MLP>MLP> MLP>Logit |

[Table 7: Iteration Summary]

As we can see from *Table 7*, all of our key metrics increased over the three iterations. Although I was not able to quantify each source of the improvement, it seemed from observation that using cross validated searching and adding phenotypic information contributed most to the difference between the first and the second. Even though the atlas type was changed from MSDL to HarvardOxford, throughout my experimentation, I realized that HarvardOxford and MSDL are remarkably similar and thus, the choice of atlas didn't make a big difference. Additionally, equalizing the data would not have been a large source of improvement in the metrics- rather this was done to combat the fact that neural networks are traditionally thrown off by imbalanced datasets (the ADHD200 is roughly two thirds non ADHD and one third ADHD). As a result,

equalizing the data might have allowed the neural networks to generate better results. Even though the Matthews Correlation Coefficient wasn't calculated for the first iteration, it is a fair assumption to make that it would improve upon in the second iteration.

Between the second and third iterations, the major changes were the switch from neural networks to randomized search on gcForest and the inclusion of phenotypic information as a feature. The gcForest model, as we can see, has a significant positive impact on non ADHD F1, MCC and a slight impact on ADHD F1. One possible reason for this is the fact that gcForest is said to be an alternative to deep neural networks that performs well even with small amounts of data. [Zhou, Zhi-Hua & Feng, Ji., 2017]. The randomized searching algorithm applied to the gcForest also could play a role in the high result attained as it allows all possibilities (within reason) to be explored. The layer structure of the best average result, MLP > MLP > MLP > Logistic Regression is interesting as it incorporates the objectively best result from the previous iteration into an ensemble classifier-- creating an extremely powerful meta-estimator. Logistic regression, however was one of the worst models in the first iteration, so the return of it into our gcForest meta estimator comes as an interesting surprise.

Rather than using a deep neural network (which might not be extremely well suited for this task given the small amount of data), which take a lot of processing, financial resources (GPU instances on EC2 are not cheap) and time to run, gcForest provided to be an effective alternative—delivering good results in a fraction of the time.

However, there has been some prior speculation as to the legitimacy of the gcForest algorithm, with prior researchers arguing that the classifiers and data sets used as the baseline used in the gcForest papers for comparison were outdated:

> *"It [gcForest] could turn out to be extremely good. However, I would say that they haven't proven their point. You can see on the test datasets, the performance is not much better than classic algorithms like SVM. I think they will need to use more difficult image datasets, where classic algorithms do not have a chance against deep learning."*
> ["Travers" - Deep Review, 2017].

We have found in this study that gcForest can prove to be an effective algorithm, especially when coupled with randomized searching, and systematic exploration of the optimal cascade structure and hyperparameters. As a side note, the gcForest authors, in their paper did say that gcForest has many less parameters to tune than a neural network, which I didn't find to be the case. In fact, the number of parameters to tune led to my need for a randomized searching algorithm.

While I did find that using phenotypic features improved model performance substantially, there were some interesting choices to make around how best to incorporate phenotypic features into the models. Initially, I felt that using 0's to equalize the matrix size between the phenotypic and functional connectome information would lead to better model results, so this was the approach I was using. However, a machine learning expert I consulted, Mr. Anton Bossenboek, shed some light on the fact that surrounding phenotypic information with 0s might confuse the model and not help it. Nevertheless, I was surprised to find out that after removing the zeros around the phenotypic information, the MCC, F1 and Precision were all lower by an average of 5%. The recall increased by 1%, which wasn't significant enough to prove that removing the zeros was a good idea. Therefore, I left the 0's surrounding the phenotypic information in the feature data in order to ensure that the phenotypic input data matrix conformed to the size of the functional connectome matrix.

**Evaluation of results versus initial hypothesis**

The best result from the ADHD200 Consortium Challenge was Johns Hopkins University. By deriving their confusion matrix from their results and comparing it to our mean confusion matrix, I was able to see how my model compares. It is important to note that my testing dataset changed with each cross validation run to a random 20% of the original dataset. This means that the number of subjects in the testing data, for us, is slightly more than in the ADHD200 supplied testing dataset.

**Johns Hopkins University**

| | True | False |
|---|---|---|
| **True** | 12 | 6 |
| **False** | 45 | 92 |

**gcForest Optimal #5 (mean)**

| | True | False |
|---|---|---|
| **True** | 25 | 9 |
| **False** | 39 | 105 |

[Tables 7, 8: Johns Hopkins and gcForest Optimal #5 (mean) confusion matrix]

From these confusion matrixes, we were able to find their precision, recall, F1 and MCC and compare it to ours.

| Metrics | Johns Hopkins University | gcForest Optimizer #5 | Improvement |
|---|---|---|---|
| **ADHD Precision** | 0.67 | 0.75 | **+0.08** |
| **Non ADHD Precision** | 0.67 | 0.72 | **+0.05** |
| **ADHD Recall** | 0.21 | 0.4 | **+0.19** |
| **Non ADHD Recall** | 0.94 | 0.93 | -0.01 |
| **ADHD F1** | 0.32 | 0.52 | **+0.20** |
| **Non ADHD F1** | 0.77 | 0.81 | **+0.04** |
| **Matthews Correlation Coefficient** | 0.22 | 0.39 | **+0.17** |

While Johns Hopkins University's non-ADHD recall is still 0.01 better than ours (93% vs 94%), we were able to improve noticeably in every other category. Specifically, ADHD F1, ADHD Recall and Matthews Correlation Coefficient are around 0.2 higher with the gcForest model. Even though more repetition of this experiment could be done for further validation and whilte there are certainly areas of improvement, it seems as that my objective to improve on prior studies was met.

My other hypothesis from Iteration 1 about how random forests and gradient boosting would provide the best models was somewhat correct, as they did provide some of the best models when used in combination as the layers of the gcForest.

**Potential sources of error and bias**

Even though gcForest seems to provide a stable and robust alternative to diagnosing ADHD, there are some possible sources of error and inconsistency within our study. Two key possible sources of error for our experiment are source validity and over diagnosed labels.

The sites that the data came from sometimes opened up possible sources of error. Some source sites for the data like NYU gave much lower results in the original ADHD200 competition, as we can see below in table 5. As an experiment, I removed the NYU portion of the dataset from the 3rd iteration optimizer and reran it. Surprisingly, it gave a much higher prediction accuracy (80%), but lower F1 and MCC, which are more rigorous metrics for this comparison. Nevertheless removing other sources with noise, such as Pittsburgh and Brown, might have allowed for an increased F1 and MCC, just like NYU did for accuracy.

Table 3: Prediction Accuracy

| | Chance[1] | Prediction Accuracy[2] | Performance Increase[3] |
|---|---|---|---|
| Overall | 38.75 | 49.52 | 10.77 |
| NYU | 32.33 | 35.19 | 2.86 |
| Brown | 37.50 | 39.74 | 2.24 |
| Pittsburgh | 38.89 | 40.74 | 1.85 |
| Peking-1 | 38.50 | 51.05 | 12.55 |
| NeuroImage | 39.00 | 56.95 | 17.95 |
| KKI | 43.18 | 61.90 | 18.72 |
| OHSU | 46.21 | 65.37 | 19.16 |

[Table 5: Chance, Prediction Accuracy and Performance Increase for each source]

Over diagnosed labels is also a potential source of error or this experiment. As I discussed in my introduction, ADHD overdiagnosis is a very predominant issue. Unfortunately, as I found while discussing with Mr. Haomiao Huang, this same issue has a good chance of impacting the validity of our data. The patients who were given the label: "ADHD" were diagnosed using the same subjective methods and processes that we are trying to improve in this study, opening up a significant potential avenue for inconsistency in this experiment. However, using unsupervised learning techniques might help with creating a more effective model in the future—using the labels as guidelines, rather than as true labels.

**Future directions**

The most obvious area for improvement in the future would be for researchers to repeat this study to further test the robustness of the gcForest models proposed in this paper. Doing this would certainly require improvements in the process of finding optimal results from the randomized searching algorithm. To find optimal results out of the ≈ 10,000 results generated over 5 different computers (two of which are on Amazon Web Services EC2 Instances), I imported the data into a Python Spark RDD. What I did not do, however was take the standard deviation (or variance) of the results across the 5 cross-validation runs into account, which might have thrown off what the best results actually were. Furthermore, the Randomized Searching algorithm was inefficient- it took 2 days to return 2000 results (per searcher). Recently, after conversing with Mr. Bossenboek, I learned of optimizers such as *TPOT* and *auto-ml* that, through genetic algorithms (based on evolution) can find optimal hyperparameters more efficiently than the random searching algorithm I was using. By using these optimizers, either on simpler ensemble classifiers like xgboost and random forests or extending the implementation to work with gcForest, better results might be found.

Additionally, adding new inputs to the model, such as doing feature extraction from the also provided sMRI volumes could improve the model. As I learned from Dr. Zhang, sMRI

volumetric analysis using Freesu mbvhkfg ik6rfer or SPM can provide a very useful feature. In addition, using more phenotypic information (medication status, IQ), while still accounting for the blank values in the CSV file might provide sources of further improvement.

Finally, even though neural networks didn't provide better results than gcForest, there are a lot of additional advanced neural networks such as Extreme Learning Machines, Long Short-Term Memory Networks and Convolutional Neural Networks which I didn't explore in the course of this experiment that should be explored by researchers in the future.

# Works Cited


Allison. "Machine Learning Part 2: Visualizing a Decision Tree." *Tech Trek*, 22 Apr. 2016, www.techtrek.io/machine-learning-part-2-visualizing-a-decision-tree/.

Benyamin, D. (2012, October 11). A Gentle Introduction to Random Forests, Ensembles, and Performance
Metrics in a Commercial System [Blog post]. Retrieved from CitizenNet website: http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics

Bossenbroek, A. (n.d.). [Personal interview by the author].

Deep Belief Networks. (2018, April 4). Retrieved April 26, 2018, from Wikipedia website: https://en.wikipedia.org/wiki/Deep_belief_network

franck-dernoncourt. (n.d.). Minimum number of layers in a deep neural network [Online forum post].
Retrieved from Stack Exchange: Stats website: https://stats.stackexchange.com/q/229659

Gorgolewski K, Burns CD, Madison C, Clark D, Halchenko YO, Waskom ML, Ghosh SS. (2011). Nipype: a
flexible, lightweight and extensible neuroimaging data processing framework in Python. Front.
Neuroinform. 5:13.

Hoogman, M., Bralten, J., Hibar, D., Mennes, M., Zwiers, M., Schweren, L., & Frank, B. (2017). Subcortical brain volume differences in participants with attention deficit hyperactivity

disorder in children and adults: A cross-sectional mega-analysis. *The Lancet*, *4*(4). https://doi.org/10.1016/S2215-0366(17)30049-4

How strong are the magnets in an MRI machine? [Blog post]. (2001, August 9). Retrieved from howstuffworks.com website: https://science.howstuffworks.com/question698.htm

Huang, H. (2018, April 4). [Telephone interview by the author].

Maini, V. (2017, August 19). Machine Learning for Humans, Part 2.2: Supervised Learning II [Blog post]. Retrieved from Medium website: https://medium.com/machine-learning-for-humans/supervised-learning-2-5c1c23f3560d

McGonagle, John. "Feedforward Neural Networks." *Brilliant Math & Science Wiki*, brilliant.org/wiki/feedforward-neural-networks/.

Monks, M. (2017, April 24). How strong is a standard magnet? [Blog post]. Retrieved from Sciencing website: https://sciencing.com/strong-standard-magnet-6853786.html

MRI interpretation Introduction. (n.d.). Retrieved from https://www.radiologymasterclass.co.uk/tutorials/mri/mri_scan

Multi-Layer Perceptron. (2018, January 25). Retrieved April 26, 2018, from Wikipedia website: https://en.wikipedia.org/wiki/Multilayer_perceptron

Notter, M. (2016). Introduction to neuroimaging. Retrieved January 5, 2018, from Nipype beginners guide website: http://miykael.github.io/nipype-beginner-s-guide/neuroimaging.html

Pedregosa, F., Varoquaux, G., Gramfort, A. & Michel, V. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Saxena, R. (2017, January 30). How decision tree algorithm works. Retrieved April 26, 2018, from Dataaspirant website: http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/

School Mental Health Project, Dept. of Psychology, UCLA. (n.d.). *Arguments About Whether*

*Overdiagnosis Of ADHD is a Significant Problem** [Fact sheet]. Retrieved March 1, 2018, from
http://smhp.psych.ucla.edu/pdfdocs/overdiag.pdf

Support Vector Machine. (2018, April 20). Retrieved April 26, 2018, from Wikipedia website: https://en.wikipedia.org/wiki/Support_vector_machine

Team, U. (n.d.). The 3 Types of ADHD. Retrieved from https://www.understood.org/en/learning-attention-issues/child-learning-disabilities/add-adhd/the-3-types-of-adhd

Travers. (2017, March 2). Deep Forest: Towards an Alternative to Deep Neural Networks #268 [Online
forum post]. Retrieved from Deep Review website: https://github.com/greenelab/deep-review/issues/268

UFLDL. (n.d.). Unsupervised Feature Learning and Deep Learning Tutorial. Retrieved April 26, 2018,
from Stanford Deep Learning Department website: http://ufldl.stanford.edu/tutorial/supervised/LogisticRegression/

Varoquaux, G., & Craddock, R. C. (2013). Learning and comparing functional connectomes across
subjects. *NeuroImage, 80*, 405-415. doi:10.1016/j.neuroimage.2013.04.007

Woodford, C. (2018, March 14). Introduction to Neural Networks. Retrieved April 9, 2018, from Explain That Stuff website: http://www.explainthatstuff.com/introduction-to-neural-networks.html

Yee, S., & Chu, T. (n.d.). A visual introduction to machine learning [Blog post]. Retrieved from R2D3 website: http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

Zhang, K., Ph.D. (n.d.). [LinkedIn interview by the author].

Zhou, Zhi-Hua & Feng, Ji. (2017). Deep Forest: Towards an Alternative to Deep Neural Networks.
3553-3559. 10.24963/ijcai.2017/497.

# Acknowledgements

This work would not have been possible if it hadn't been for the many people who assisted me throughout the and research, programming, writing of this experiment. In particular, I would like to acknowledge Mr. Sarabjit Baveja, Mr. Carl Ma, Mr. Karl Schmidt, Dr. Haomiao Huang, Ph.D., and Dr. Kai Zhang, Ph.D. for being amazing mentors and resources throughout this process.

---

*I pledge to uphold the Branson Honor Code. I have neither received nor given unauthorized aid*

# Glossary

| Term | Definition |
|---|---|
| **Python** | Python is an interpreted high-level programming language for general-purpose programming. |
| **API** | An API is a set of clearly defined methods of communication between various software components. |
| **MRI** | MRI uses a powerful magnetic field, radio frequency pulses and a computer to produce detailed pictures of organs, soft tissues, bone and virtually all other internal body structures. |
| **fMRI Volume** | Functional magnetic resonance imaging or functional MRI (fMRI) measures brain activity by detecting changes associated with blood flow. fMRI volumes are a 4D image containing the visualization from the fMRI scan. |
| **Functional Connectivity** | Functional connectivity refers to the functionally integrated relationship between spatially separated brain regions. |
| **Classifier/Estimator/Model** | In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs. A |

| | |
|---|---|
| | classifier/estimator/model is what is used to identify this. |
| **nilearn** | Nilearn is a Python module for fast and easy statistical learning on NeuroImaging data. |
| **sklearn** | Machine Learning in Python. Simple and efficient tools for data mining and data analysis; |
| **CNN** | In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. |
| **Keras** | Keras is a high-level neural networks API, written in Python |
| **Random Forests** | Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees |
| **Decision Tree** | Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). |
| **Logistic Regression** | Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. |
| **KNearest Neighbors** | In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. |
| **SVC** | A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. |

| | |
|---|---|
| **Restricted Boltzmann Machine** | A restricted Boltzmann machine (RBM) is a generative stochastic unsupervised artificial neural network that can learn a probability distribution over its set of inputs. |
| **Autoencoder** | An autoencoder is an artificial neural network used for unsupervised learning of efficient codings. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. |
| **Multi Layer Perceptron** | A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. |
| **Neural Network** | An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. |
| **Deep Belief Network** | In machine learning, a deep belief network (DBN) is a generative graphical model, or alternatively a class of deep neural network, composed of multiple layers of latent variables ("hidden units"), with connections between the layers but not between units within each layer. When trained on a set of examples without supervision, a DBN can learn to probabilistically reconstruct its inputs. The layers then act as feature detectors. After this learning step, a DBN can be further trained with supervision to perform classification. |
| **MRI Preprocessing** | Preprocessing is the term used to for all the steps taken to improve our data and prepare it for statistical analysis. We may correct or adjust our data for a number of things inherent in the experimental situation: to take account of time differences between acquiring each image slice, to correct for head movement during scanning, to detect 'artifacts' – anomalous measurements – that should be excluded from subsequent analysis; |

| | |
|---|---|
| | to align the functional images with the reference structural image, and to normalize the data into a standard space so that data can be compared among several subjects; to apply filtering to the image to increase the signal-to-noise ratio; finally, if sMRI is intended, a segmentation step may be performed. |
| **Extra Random Forests** | With respect to random forests, the method [Extra Random Forests] drops the idea of using bootstrap copies of the learning sample, and instead of trying to find an optimal cut-point for each one of the K randomly chosen features at each node, it selects a cut-point at random. |
| **Gradient Boosting** | Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. |
| **XGBoost** | XGBoost (Extreme Gradient Boosting) is an open-source software library which provides the gradient boosting framework for C++, Java, Python, R, and Julia. |
| **Phenotypic** | relating to the observable characteristics of an individual resulting from the interaction of its genotype with the environment. |
| **Confusion Matrix** | A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. |
| **F1 Score** | The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. |
| **Precision** | Precision is the fraction of relevant instances among the retrieved instances |

| | |
|---|---|
| **Recall** | Is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. |
| **Matthews Correlation Coefficient** | Matthews Correlation Coefficient. The Matthews Correlation Coefficient (MCC) has a range of -1 to 1 where -1 indicates a completely wrong binary classifier while 1 indicates a completely correct binary classifier. Using the MCC allows one to gauge how well their classification model/function is performing. |
| **Accuracy** | Accuracy is the number of correct classifications out of the total number of classifications |
| **Standard Deviation** | a quantity calculated to indicate the extent of deviation for a group as a whole. |
| **gcForest** | gcForest is a decision tree ensemble approach with performance highly competitive to deep neural networks. |
| **a feature** | In machine learning and pattern recognition, a feature is an individual measurable property or characteristic of a phenomenon being observed. Choosing informative, discriminating and independent features is a crucial step for effective algorithms in pattern recognition, classification and regression. |
| **Cross Validation** | Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it. |
| **Machine Learning** | Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) with data, without being explicitly programmed. |
| **NiFTi** | NiFTi is a file format for the storage of MRI scans |
| **Apache Spark** | Apache Spark is a unified analytics engine for big data processing, with built-in modules for |

| | |
|---|---|
| | streaming, SQL, machine learning and graph processing. |
| **RDD** | The main abstraction Spark provides is a resilient distributed dataset (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. |
| **SQL** | SQL is a standard language for storing, manipulating and retrieving data in databases. |
| **AWS (Amazon Web Services)** | Amazon Web Services offers reliable, scalable, and inexpensive cloud computing services |
| **EC2** | Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. |
| **Gini Impurity** | Gini Index is an indicator of how the classification split is with respect to the classes. |
| **Extreme Learning Machine** | Extreme learning machines (ELM) are single-hidden layer feedforward neural networks (SLFNs) which randomly chooses the input weights and analytically determines the output weights of SLFNs. In theory, this algorithm tends to provide the best generalization performance at extremely fast learning speed. |
| **LSTM** | Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. |