I. Introduction

The goal of this analysis is to model the swinging behavior of Major League hitters using machine learning techniques. The machine learning techniques will use the pitch f/x data points, combined with game scenario data points, to predict whether or not a hitter will swing at a given pitch.

Just like how Google uses Machine Learning techniques to teach a machine to drive like a human, MLB can use Machine Learning techniques to train computers how to mimic players. Being able to predict whether a hitter will swing at a pitch can open up new forms of advanced scouting. If a model can be customized to a particular player, then teams can create simulations on how to best get that player out, with the pitchers they have at their disposal.

Teams can simulate pitcher-hitter matchups that have never occurred. Teams can simulate pitch sequences to test scouting reports. Pitchers can throw simulated games against the computer model to help them practice the upcoming game plan. There are many more applications of a model like this. The question is how do we know how accurate the probability outcomes are for each pitch? Below we will outline the process and demonstrate why they are accurate.

II. Data

The data points used are:

1. **releaseVelocity** - pitch velocity converted to feet per second
2. **batterHand** – Righty or Lefty
3. **pitcherHand** – Righty or Lefty
4. **x0 -** distance in feet from the center of the plate at the first measurement (50 ft from home plate), negative values are inside to RHB
5. **y0** - distance in feet from the tip of home plate at the first measurement (50 ft from home plate)
6. **z0** - height in feet off the ground at the first measurement (50 ft from home plate)
7. **vx0** - the velocity of the pitch, in feet per second, in the x dimension
8. **vy0** - the velocity of the pitch, in feet per second, in the y dimension
9. **vz0** - the velocity of the pitch, in feet per second, in the z dimension
10. **ax** - the acceleration of the pitch, in feet per second per second, in the x dimension, measured at the initial point.
11. **ay** - the acceleration of the pitch, in feet per second per second, in the y dimension, measured at the initial point.
12. **az** - the acceleration of the pitch, in feet per second per second, in the z dimension, measured at the initial point.
13. **spinRate –** pitch spin in RPM
14. **spinDir** - from the catcher's perspective, the angle (from 0 to 360) between the the pole around which the ball is rotating and the positive x-axis
15. **manonFirst** – binary variable indicating a runner on 1st Base

16. **manonSecond** - binary variable indicating a runner on 2nd Base
17. **manonThird** - binary variable indicating a runner on 3rd Base
18. **inning** – the inning the pitch occurred
19. **count** – the balls and strikes count before the pitch is thrown.

I then created some additional variables:

20. **score_diff** – the difference in the score from the batter's perspective
21. **Pitch_In_AtBat** – the number of pitches the batter has seen in the at-bat
22. **Decision_lag1** – the outcome of the previous pitch seen in the at-bat

The dependent variable is **Decision**. The **Decision** is a binary outcome of whether the batter swings or does not swing.

III. Model

I first convert every pitch outcome that Miguel Cabrera saw in the 2014-2016 MLB seasons to either "Swing" or "Take." I then separate the data into a training set and a test set. The training data consists of all pitches seen during 2014-2015 seasons, and the test data consists all of the pitches seen during the 2016 season. After removing non-decision type pitches, like intentional balls, hit-by-pitch, Missed Bunts, Foul Bunts, Pitch Out, Automatic Strike, Automatic Ball, and Unknown, I had 4,324 pitches in the training set and 2,259 pitches to test out of sample.

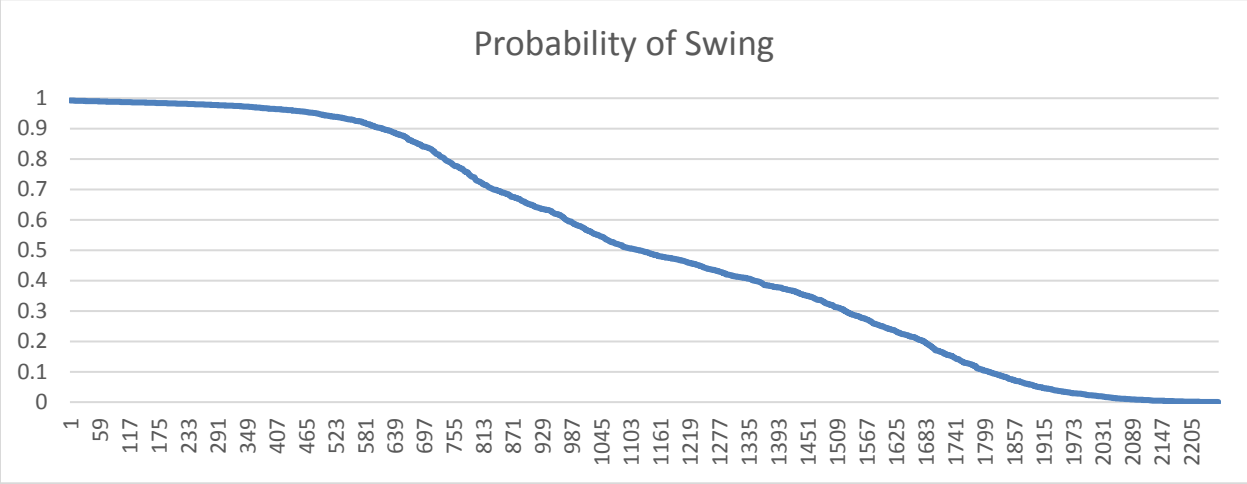
The type of model used in the analysis is a neural network model. Neural networks are useful because they can handle complex interaction effects between its X variables. After several iterations of randomly selecting parameters, the most predictive out of sample results were as followed:

		Predicted		
		Do Not Swing	**Swing**	**Error Rate**
Actual	**Do Not Swing**	880	243	0.2164 = 243/1123
	Swing	233	903	0.2051 = 233/1136
	Totals	1113	1146	0.2107 = 476/2259

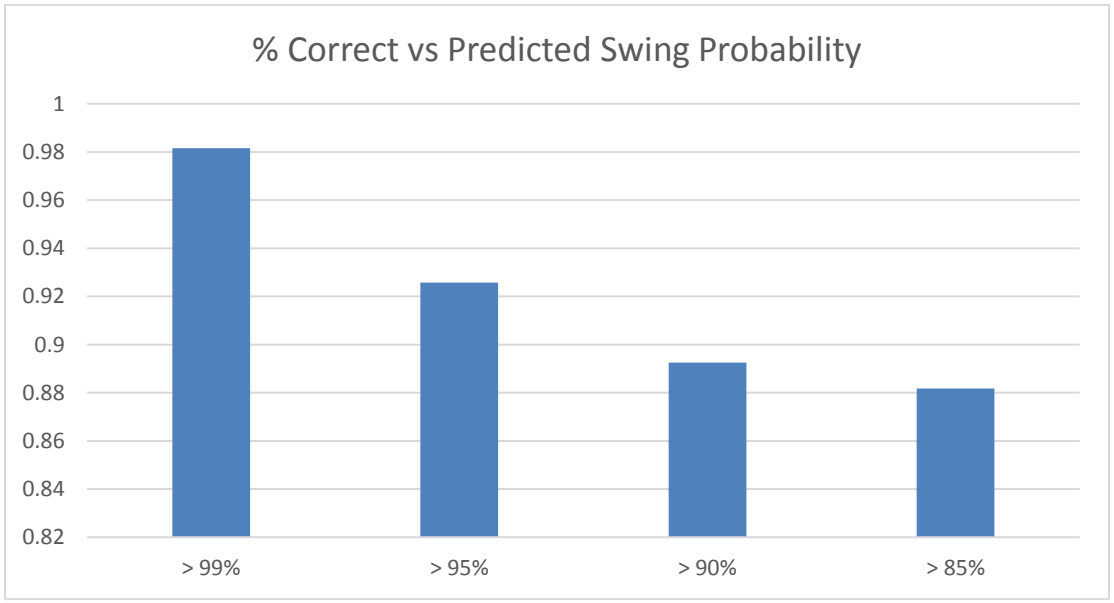
Using an optimal threshold of 0.50 for predicting Swing, the overall Error Rate of the 2259 pitches is 21.07%. The AUC of the out of sample set is 0.8714 with a LogLoss of 0.467.

Probabilities…

The below graph is of the range of Swing probabilities for each of the 2259 out of sample predictions. We can see that there are extreme predictions, where we see predictions of nearly 100% and 0% of whether Cabrera will swing.



In this next graph, we can see the percentage of correct predictions for the range in predicted Swing probabilities. For example, there were 54 pitches that have Swing probabilities greater than 99% and 53 of them were predicted correctly (98% accuracy). 485 pitches were predicted with greater than 95% probability and 449 of them were predicted correctly, an accuracy rate greater than 92%.



These are promising results. It would be problematic to see predictions of greater than 99% not be nearly 100% accurate.

Conclusions…

Overall, the model seems to do quite well out of sample.

The model is to be used as a building block. Once a model is successful at predicting when a player will swing, then we can start simulating actual events from those swings. This paper is not designed to be a final solution to predicting swinging behaviors. Instead, it is a building block for future predictive analysis.

Appendix

```
nn_model_Decision <- h2o.deeplearning(
 x=predictor_indices_Decision, y=response_index_Decision,
 training_frame=train_data_h2o_Decision,
 balance_classes=TRUE,
 activation="RectifierWithDropout",
 rate = .025,
 #input_dropout_ratio=.05,
 #hidden_dropout_ratios = c(0.6, 0.6, 0.6, 0.6),
 #classification_stop=-1,  # Turn off early stopping
 #l1=1e-5,              # regularization
 hidden=c(250,250,250,250),
 epochs=69,
 model_id = "NeuralNet_MNIST_001",
 reproducible=TRUE,
 seed=99,
 export_weights_and_biases=TRUE,
 ignore_const_cols=FALSE,
 distribution = "bernoulli",
 variable_importances = TRUE,
 keep_cross_validation_predictions = TRUE)
```