A Comparison of Machine Learning Approaches to Horse Race Handicaping

A Big Data Centered Retirement Plan

James Spiker

University of Nevada, Reno

Reno, Nevada, U.S.

jspiker@nevada.unr.edu

Nathaniel McFadden

University of Nevada, Reno

Reno, Nevada, U.S.

nmcfadden@nevada.unr.edu

*Abstract*—Horseracing has been a popular sport with the public and with gamblers for hundreds of years. The relatively few variables that contribute to the success of a horse and the availability of data that quantifies its performance in a race lends itself well to a computerized handicapping model. This paper compares two such techniques and their effectiveness. We used readily available statistics to develop an ordinal multinomial logistic regression model and a neural network model. While both models were simplistic, comparison of the results show that the multinomial logistic model matched the performance of the public model, and the neural network model outperformed the public model when picking the winner of a race over the course of a season.

# Introduction

Horseracing has been a past time for mankind for hundreds of years. As such, so has gambling on outcomes of a horserace between spectators. To this end, developing a model that gives one bettor an advantage over another, has been the goal of gamblers for years.

Handicapping is the process of taking available information and using it to choose the winning horse in a race. Information on past performances, bloodlines, training regimens, starting draws and many other factors can all contribute to a horse’s ‘value’ in winning and losing. Models developed for handicapping a race will be dependent on the person building the model. There is no ‘one’ right way to pick a winner. Using information available from published ‘race cards,’ gamblers attempt to determine who is most likely to win a race based on statistics like ‘race length’, ‘speed rating’, placing history, etc.

With the advancement of technology, handicapping horseracing has become more codified. Bolton and Chapman were among the first to use ordered multinomial logistic regression to bring mathematical rigor to the handicapping process, with “Searching for Positive Returns at the Track: A Multinomial Logit Model for Handicapping Horse Races.” [1] They used data compiled from 200 horse races to get promising prediction results using a multinomial logit model.

William Benter built on Bolton and Chapman [2], using the idea of a Multinomial Logit model to earn close to 1 billion dollars during the 1980’s and 1990’s in Hong Kong. Benter used a team of analysts to enter the data from previous races to build the database for his model. Today Benter runs a hedge fund centered around his horse betting algorithm [3]. This algorithm is not public.

Since Benter deployed his model, technology has continued to advance. While not a new concept, society lacked the computing power to employ neural networks until more recently. However, in the last 15 years or so, computing power has reached a point where implementing neural networks is commonplace. The somewhat deterministic nature of horse racing makes it a great application for a machine learning model. This paper builds horse racing models using both techniques and compares the accuracy of the results.

Horseracing is a sport that is popular in markets around the world. However, variables such as race distance, jockeys, tracks, weather, and elevation change from market to market. As such, there are no guarantees that a model developed with data from one market will necessarily work in other markets. With this in mind we sought a market that had a high number of races, a fairly closed pool of horses to choose from, and had data on the races readily available. The Hong Kong Jockey Club fulfilled these three requirements.

# Ordinal Multinomial Logit Model

## Processing Data for the Multinomial Logit

Initially, we attempted to scrape the publicly available data from the Hong Kong Jockey Club’s website. All data from all past races and all horses are accessible from the website. When we attempted to access the data, however, the website employed CAPTCHA modules to prevent the automated scraping of their information. After much effort, we decided to purchase the dataset an individual who had already scraped the information.

The dataset itself was a .CSV file roughly 330 megabytes in size. The data was grouped by race, with each race having a unique ID number, with each horse in the race also having a unique ID number. Each entry contained a total of 55 separate pieces on data on the horse. The entire file was read into a data frame with Pandas for processing and organization.

We divided the dataset into two groups. A training set and a testing set. The training dataset consisted of the 2019, 2020, and 2021 racing seasons. This consisted of roughly 2500 races, and 25000 individual horses racing. This provided ample information to build the horse profiles needed for prediction. The testing data consisted of the 2022 racing season up to March 27th.

Pandas was extraordinarily important in the efficient processing of the data from the ordinal multinomial logit model. Because each race needed to be considered on its own in order to predict its winner, we needed to sort through thousands of lines of data to find all information available on the horses in that particular race. Pandas was able to do this easily. This allowed us to code profiles for each horse in the race and use those profiles to quickly develop deeper and more explanatory statistics for our model. Using Pandas we built statistics for each horse like, lifetime win percentage, time since last race, trainer, jockey, and sire winning percentages, and average speed rating. Once we had a preliminary list of statistics, we thought my factor into the model we produced a correlation matrix to determine with parameters had the strongest effect on place.Chart, treemap chart

Description automatically generated with medium confidence

Fig. 1. Correlation Matrix for Preliminary Statistics

Based on the correlation matrix, we chose the following parameters to enter into the ordinal multinomial logit model.

*Speed Rating*: Speed rating is a statistic that measures and normalizes the speed of the horse compared to the other horses in the race. Race organizers use the statistic to keep races competitive. This statistic was available in the data set.

*Days since last race*: The number of days since the horse last raced. To build this statistic we used pandas to find the date of the last race and then calculate the number of days that have elapsed.

*Winning Percentage*: How often the horse wins. This was calculated by summing the total number of wins in a horse profile and dividing by the number of races it has run.

*Place Last Race/ Place Second to Last Race*: The places of the horse in the last two races. This was also derived using Pandas, by searching the horse’s profile and returning the respective places.

*Sire Strike Rate/Trainer Strike Rate/Jockey Strike Rate*: The win rates of the horse’s sire, jockey, and trainer. These were calculated with Pandas, by retrieving the name of the trainer, jockey, or sire, then searching all the data for those entries and calculating the win percentages of those individuals.

*Public Probability*: The closing odds of the horse to win the race is part of the dataset. These odds represent the public’s estimation of the chances of a horse to win the race. This data was of the form, 2 to 1, or 40 to 1. We used Pandas to convert these odds into probabilities.

The statistics used for the model are by no means advanced, or to be considered exhaustive or complete. In fact it should be assumed that these statistics are very naïve, and should only be looked at as a starting point for this model.

## Ordinal Multinomial Logistic Regression

In “Searching for Positive Returns at the Track: A Multinomial Logit Model for Handicapping Horseraces.” Ruth Bolton and Randall Chapman laid the groundwork for a rigorous mathematical approach to predicting winners of the race. Prior academic research in this field had focused on betting systems to increase returns, as well as examinations of the public odds model of handicapping, otherwise known as simply choosing the favorite.[1] The multinomial logit model is ideal for handicapping because it recognizes that there is only one winner in each race. The logit model treats each place as a categorical variable, and then uses the log function to force the calculated probabilities to the nearest category. If a probabilistic regression is given by the following equation,

P(Y=1|X1,X2,…,Xk)=F(β0+β1X1+β2X2+⋯+βkXk) (1)

And then applied to the following CDF:

(2)



We are left with the following equation for multinomial logistic regression:

(3)



Where Xn is a parameter of the model, and βn is the weight attached to that parameter in the model.

## Ordinal Multinomial Logistic Regression Results

This model produced no clear information above and beyond what the public (favorite) model produced. Of the 537 races in the testing set, the multinomial model correctly predicted the winner 167 times, or 30% of the time. The public model correctly predicted the winner in 170 races, 30.5% of the time. There were 15 races where our model predicted the race winner correctly and the public model did not. There were 18 times when the public model predicted the correct winner when our model did not. The 33 races where one of the models picked the correct winner and the other did not were races in which the probabilities of the picks by the models differed by less than five percent. And while the difference in results is not statistically significant, it is interesting to note how our model performed compared to the public model with respect to return on investment. If a fifty-dollar bet was made on each horse picked to win in each race of the season, our model would have earned over a 500% return. While the public model was also profitable, out model was able to pick winners that had higher odds and therefore, higher returns.

Chart, line chart, histogram

Description automatically generated

Fig. 2 Profits for the Public Model v. Multinomial Logit Model

## Next Steps

When the predicted probabilities are close to the public model probabilities, the logit’s success rate was no more than a coin flip. Benter also ran into a similar problem when running his model. His solution was to run a second logistic regression between the public probabilities and the predicted probabilities. This served to create greater differentiation between the two models and did so to his model’s favor. Implementing this statistic would be an obvious next step. Additionally, refining and trying more sophisticated statistics in the model will also produce more accurate results.

# Neural Network Model

## Preprocessing Data For Training

The same dataset utilized for the Multinomial Logistic Regression was applied when designing the Neural Network. However, the process of modifying and processing the data differed greatly. Pandas was used to create a data frame that containing a total of seventeen columns. Each column represents an important variable relating to each race.

*Race ID:* The Race ID refers to the unique number that identifies each race. This column was used for indexing to provide structure to the data.

*Racetrack:* The racetrack refers to the specific track that was the host for the current race.

*Race class:* The raceclass represents the category for the current race. These categories are representative of the types of horses and Jockeys that are racing. Usually, they are divided by ability and track types.

*Track type:* The track type refers to the specific shape and configuration of the racetrack.

*Distance:* The distance refers to the length of the track in meters.

*Going:* The going of the racetrack refers to the conditions of the turf the day of the race. These conditions can be represented by the following ratings: Heavy, Soft, Good to Soft, Good, Good to Firm, or Firm.

*Course:* An alternative designation for the racetrack. This was included just to ensure the data was accurate. As if the course and track differed, there was some error.

*HorseID:* The horseID refers to the numerical classifier that distinguishes each horse. This acts as a numerical alternative to the horse’s name.

*Jockey:* The Jockey is the racer riding the horse. They are distinguished by their names.

*Trainer:* The trainer refers to the individual that trained the horse that is being rode by the Jockey.

*Sire:* The sire refers to the father of the horse.

*Country:* The country refers to the place of origin for the horse.

*Draw:* The draw refers to the specific position the horse started the race at. The first draw is the closest to the inside while the fourteenth draw is the farthest from the inside.

*Agecurrent:* The “agecurrent” refers to the age of the horse the date of the race.

*Declaredweight:* The “declaredweight” refers to the weight of the horse in pounds the day of the race

*Actualweight:* The “actualweight” refers to the combined weight of the horse, jockey, and their equipment the day of the race

*Winodds:* The “winodds” refer to the current odds of each horse winning the race. These odds are dictated by the public and where money is being placed.

*Place:* The place of the horse refers to the finishing position of the horse.

After the data columns were decided, the next step of the process was the encoding of data that had inconsistent data types and the entries of data that were structured as strings. When training a model with Tensor Flow and Keras, it is important that all data is in numerical data types such as integers and floats. Boolean values are also accepted.

There were two types of encoding implemented in this project. These two types of encoding include nominal encoding and ordinal encoding. The main difference between these two types are how they organize their data. Nominal encoding does not order the data in any specific way, as there is no relation between the entries in the data. Ordinal encoding organizes the data in a hierarchal structure, where certain values are considered higher or lower than other entries.

Encoding the data was simplified using the scikit learn library. As such, the process of encoding each data entry was simplified by making a separate encoder for each column of data. Each encoder than modified the values in that column and then used the fit\_transform function to restructure the new data to fit within the existing structure of the data frame. Each piece of data encoded was encoded if it was in string format, or if the data types were inconsistent within the same column.

*Ordinal Encoding:* The pieces of data that were encoded according to an ordinal scale include: the going, course, racetrack, and track type.

*Nominal Encoding:* The pieces of data that were encoded according to a nominal scale include: the horse ID, jockey, trainer, sire, country, and raceclass.

After the data has been encoded to either integers, floats, or Booleans. The data must be standardized to reduce the influence of each piece of data being represented by different units. For this purpose, sickit learn was utilized again. The function “preprocessing.StandardScaler()” was used to scale the data. The formula used by this function is shown below.

(4)

The various variables are defined as follows: z represents the newly standardized sample, x represents the sample before standardization, u represents the mean of the samples, and s represents the standard deviation of the training samples. After the standard scaling, there is one final step to the preprocessing stage of the Neural Network.

When analyzing the data, one aspect of it becomes very apparent. The data is severely imbalanced. There are thousands of races with up to fourteen horses entered into every race. There is only a singular horse that can be the winner of a race, so there is up to thirteen losers per race and only one winner. Therefore, the majority class is represented by horses that lose the race. This imbalance creates a problem. Originally, this problem was ignored. The results lead to the system simply classifying every horse as a loser. While this strategy was highly accurate, the results were useless for the purposes of handicapping. To fix this problem, a sampling strategy was applied.

SMOTE, or Synthetic Majority Oversampling Technique, refers to the process of creating synthetic duplicates of the minority class. This results in copies of data entries being made to equalize the population of both classes. This process does not give the system any new information, but it incentivizes the Neural Network to guess that a horse is a winner.

## Training and Model Construction

The preprocessed data was split into four different segments. Training and testing for the input and training and testing for the output. The data entries used for training and testing were randomized, but the data had a split of eighty percent training and twenty percent testing.

This model is a Logistic Regression based on binary classification. As such, there are two possible outputs for every row entered into the model. The model can output either a one for winning or a zero for losing. The model is structured as follows: an input layer that accepts seventeen inputs, a hidden layer with eight neurons, a dropout layer that drops twenty percent of the data, a hidden layer with six neurons, and finally an output layer that returns a value that is either one or zero. The system uses the Adagrad optimizer which is one of many available optimizers from the Keras library. This specific optimizer is most effective when paired with categorical or binary classification models.

Diagram

Description automatically generated

Fig. 3. Neural Network Model

After creating the model, the only step left was to run the testing and training simulations to see how our model works when the data is entered into the simulation. For clarification, the final layer is a sigmoid output. Sigmoid outputs result in values between zero and one. Values are then rounded according to standard rounding rules. If the value is above 0.5 it is rounded to 1. Values below the 0.5 threshold are rounded to 0.

## Neural Network Results and Analysis

There were two full runs of the neural network that were recorded. The first run is without the application of SMOTE. As such, the results differ greatly from the second set of results.

Chart

Description automatically generated

Fig. 4. Validation Precision/Accuracy Run 1

Figure 4 refers to the precision or accuracy of the model’s prediction. This measures how often the model was able to correctly guess the outcome of a race. It is measured in percentage of guesses that are correct over the measure of epochs or runs through the data. Each epoch represents one entire run through the data by the model. This means each run of the program tests the dataset two-hundred times.

As we can see, the model seems to be incredibly accurate; however, there is one issue here. The model is only accurate since it always classifies the horses as losers. This works out for the validation accuracy since most horses are going to lose the race, they participate in. However, as stated earlier, this model is not efficient at making profits from betting. Betting on horses losing does not make money, to make a profit, the model needs to predict winners correctly.

Graphical user interface

Description automatically generated with low confidence

Fig. 5. Validation Loss Run 1

Figure 5 refers to the model’s loss, this measure determines the percentage of loss over the epochs. The loss refers to how confident the model is in its predictions. The lower the percentage loss, the better. As we can see, the model seems to perform exceptionally. Unfortunately, there is the issue of the data being imbalanced. Therefore, the results are skewed and not completely accurate.

Chart, line chart

Description automatically generated

Fig. 5. Validation Precision/Accuracy Run 2

Figure 5 refers to the precision/accuracy of the model. Due to the implementation of SMOTE, the results seem far less promising than the original model. This is a necessary sacrifice, as there was no way to implement the first model into any betting algorithm or scheme. This model can predict the correct outcome around 56-57% of the time. This is an impressive metric that seems to have room for growth.

A picture containing histogram

Description automatically generated

Fig. 6. Validation Loss Run 2

Figure 6 represents the loss of the model during the second run where SMOTE was implemented. This shows a steady decline in loss. However, there is still a large amount of loss recorded by the end of testing and training. There seems to be a loss of nearly 60% by the end of the simulation. This is higher than we would like. However, again, the sacrifice had to be made to ensure the model would select horses as winners.

The model seems to reach a significant amount of accuracy in its predictions. However, there is still a large amount of room for growth. Further testing and modifications must be conducted before the model can be taken out to market.

# Conclusion

## Model Comparision

When comparing the effectiveness of both models, there are a few key points to reference: the precision and accuracy of the models, the room for improvement, and the possibility for profit.

The Neural Network seems to have a higher degree of accuracy regarding the prediction made towards the winners, as such it seems to be the superior model in this specific metric of comparison.

Both models are incredibly naïve. They incorporate a proportionally small amount of data. Therefore, both models leave a large amount of room for improvements to their complexity and depth of knowledge. Many variables from the original CSV file were dropped for time’s sake. There are also issues regarding the handicapping of horses with higher odds. In the races conducted by the Hongkong Jockey Club, there is a system put in place that adds additional weight to horses that seem to have higher odds of winning. As such, an entire metagame regarding the class of race and the odds on the horses in each class has surfaced. This leads to jockey’s purposely losing races to lower their class in hopes of creating larger winning streaks where they are forced to carry less weight. Neither model has accounted for this phenomenon.

Both models seem to have potential regarding making a profit. However, only the Ordinal Multinominal Logistic Regression model has been fully tested with a simulation outfitted towards betting. As such, it is the only model of the two that is safe to bring out into the market until further testing has been conducted.

## Summary and Future Plans

Considering the reasonable amount of success both models have displayed regarding the handicapping of horses, there is merit in continuing the development of an inclusive model for betting. To continue development, combining the two models into one singular model seems like the next step towards creating the most accurate model possible. Benter originally used a logistic regression between the public model and his own to truly differentiate his from competitors. A similar approach can be implemented here by incorporating the data from the Ordinal Multinominal Logistic Regression model into the Neural Network. By adding the newly calculated odds into the model, the model should perform with greater accuracy. However, further testing is require to confirm these suspicions.

##### References

1. R. Bolton, R. Chapman, “Searching for Positve Returns at the Track: A Multinomial Logit Model for Handicapping Horseraces,” Management Science, vol. 32, p. 1040, Aug 1986.
2. W. Benter, “Computer Based Horse Race Handicapping and Wagering Systems: A Report,” World Scientific Book Chapters, in: Donald B Haush & Victor SY Lo & William T Ziemba, Efficiency of Racetrack Betting Markets, chapter 19, pages 183-198, World Scientific Publishing Co. Pte. Ltd..
3. K. Chellel, “The Gambler Who Cracked the Code.” Bloomberg.com. Retrieved May 6, 2022, from https://www.bloomberg.com/news/features/2018-05-03/the-gambler-who-cracked-the-horse-racing-code