Cornell University

ORIE 4741 - Final Report

Professor Udell

Exploring Expected Value with Horse Race Betting

Anthony Peraza (atp44), Warren Blood (wmb87), Swastik Chaki (sc2267)

December 14, 2020

Table of Contents

0.1 Abstract

The Hong Kong Jockey Club provides a dataset of 6,349 races from 1997 to 2005, with each race consisting of 14 horses with 37 attributes per horse. The ubiquity of Horse racing in sports betting communities provides and interesting opportunity to apply machine learning techniques for predicting the correct horse to place bets on for the maximum payoff. Multiple different modeling techniques were explored and the results show that a Random Forest regression model has most promising metrics. Finally we discuss, potential strategies with these models and limitations.

0.2 Introduction

Background Information - Horse Racing Basics

Typical horse racing consists of different lengths tracks, i.e. 1000m, 1650, etc, where a group of 14 horses, compete to win. Each horse is ridden by a "Jockey" who is the person actually riding atop the horse during the race. For a given race, the track will take bets on each horse ranging from simple 'win bets' - "horse #7 to win" - to complex parlays, where multiple conditions have to be met for the betting party to be successful. The majority of these betting systems are whats known as "Parimutuel Betting" where all bets, of a particular type, are pooled together, the 'house' deducts its fee and the payoff odds are calculated by sharing the pool among the winning bets. For the purposes of our project, we will be focusing on the payouts from simple "win bets". Finally, a very important aspect of horse race betting is that betting is allowed until the gun goes off for the race. This means that the odds for each horse often change up to the start time.

Aspirations

With this dataset from the Hong Kong Jockey Club, we hope to answer the question: "Can we find a positive expected value to generate a profit by predicting a probability of a given horse winning its race?" Of course, we don't just want to bet on the horse that's most likely to win, we want to bet on the horse with the highest expected payoff for a given race. We aim to accomplish this by generating two sets of models, one to predict the probability of a horse winning and one to predict the expected payout if the horse did win, i.e. win dividend.

0.3 Exploratory Data Analysis

0.3.1 Data Characteristics

Our data began as two distinct CSV's, one detailing the 6,349 different races over the 8 years and another detailing about 80,000 horse performances. After running an inner join on the two datasets using the *race_id* column, we had 66 features per row.

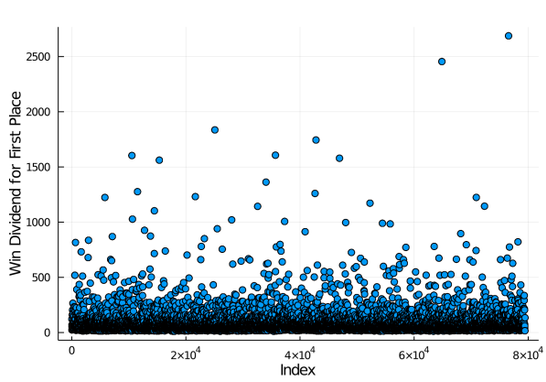
Following intuition, our data will be inherently skewed. Since we have about 6,000 races and 80,000 performances we will have a majority of losses for the horses. Thus our data will be unbalanced with respect to winning horses. This we will discuss farther into our preprocessing pipeline through the use of a balanced error metric.

0.3.2 Data Cleaning

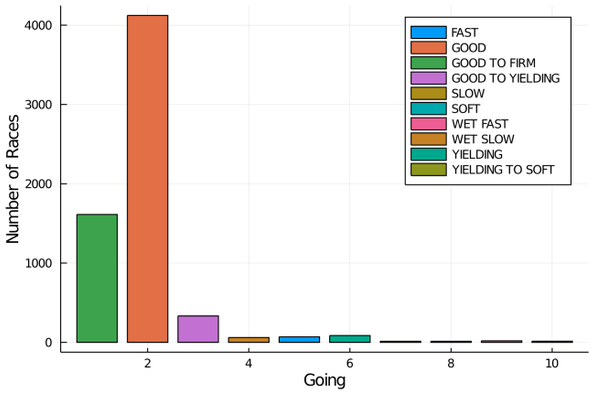
Our data had multiple 'messy' issues ranging from categorical columns and dealing with dates to totally inconsequential data and corrupted values.

Unnecessary Data The first processing task we decided to tackle was removing insignificant features. By specifically defining our problem to solve, Our data included a significant amount of unnecessary columns. For instance, for our task of predicting race outcomes (horse winning or not) we do not have use for any feature regarding events within the race. This would be all of the splits (i.e. specific lap time for a horse) and the section times (time for last horse to finish lap). Following the same logic, since our project only focuses on the winning horse, the columns that recorded the 2nd, 3rd, and 4th finisher and its win dividends will subsequently removed. After removing the unnecessary columns, our dataset had a remaining 22 features.

Outliers We found a very significant skew in our second target column: *win_dividend1* (see Figure 1a) . The mean of the column was 97 with a median of 59. However, the max of *win_dividend1* was 2687 where the min is 10.5. This means we have an extremely small portion of races that are most likely 'high-profile' events such as the most famous races of the year etc. For this project we will not be trying to predict those very high profile events since the variability will undoubtedly be higher. We will be focusing on the 'normal' races that happen through the season of a year which means we can remove some of these extremely high *win_dividend1* entries. We selected a threshold of 750, keeping every entry below, because the rows that we removed were just about 1% of the total amount of rows. Where we assume only 1% of the races per year are 'high profile' events.



(a) *win_dividend1* Outliers



(b) An example of skewed categorical feature.

Figure 1: Outlier Presence and Uneven Categorical Feature

Encoding and Data Transformation Now that we have relevant data without significant outliers we can begin manipulating our categorical data. We had 8 categorical features which were one-hot encoded. After this process we had added 436 new features to our dataset, making it very sparse. Many of our categorical features are unbalanced (see Figure 1b), especially, two of our columns, *jockey_id* and *trainer_id* had 186 and 176 distinct values respectively. This intuitively makes sense because the jockeys are the different professionals riding the horse and it was surprising to us there were only 186 different people over 8 years. This shows that jockeys and horse trainers stay in their industry for a significant time frame and therefore, we thought, it was important to keep the sparsity since it represents different people with unique impactful skills.

Another aspect of our data we had to transform was the date column. The date feature were originally represented as Date objects which could not be directly passed on to model calculations like Least-Squares because it is not a float/int. Therefore, we converted the Date objects into a day of the year so our model could still account for seasonality effects. Our final transformation was imputation of the prize column. The prize column represents the winnings that the owner of the horse and jockey will split. This could shed light on the significance of races with obviously higher prize pools at high profile events. To handle the missing values within these column we decided to impute the mean since we found the prize column was approximately a normal distribution.

0.4 Model Selection

0.4.1 Win Dividend

There are two parts to calculating an expected payoff. One is the win probability of a specific horse, which we will discuss later. The other part is to multiply this expected win probability by amount of money that the winning horse gets. The product of these two quantities gives us the expected payoff. In our data set, the Win Dividend column tells us the amount of money that the winning horse receives. Thus, we will try to find a model that predicts the final win dividend of a race with the lowest MSE.

We first constructed a least squares regression model to have a baseline MSE to which we could compare future models to. Our intuition in fitting a least squares model is simple. The variables in our data set are somewhat likely to have a linear relationship with the total win dividend. The objective function of the least squares model is defined as the following:

$$\text{minimize } \sum_{i=1}^{n}(y_i - w^T x_i)^2$$

After fitting the least squares model, we discovered that linear regression may or may not be the best model to fit based on the MSE. An interesting revelation that we came across is that there are a number of very high outliers in our data set that are causing the MSE to be unusually high. This is why we decided to also compute Mean Absolute error to get a clearer picture of the model's accuracy. This will be discussed further in "Model Results."

Least Squares essentially served as a default model to which we can compare future models to. Moving forward, we decided that because linear regression tends to over fit sometimes, we should choose a model that reduces over fitting in order to balance it out. Additionally, as previously mentioned, we suspect there may be large outliers in our data that is causing the MSE for least squares to be unusually high. Thus, we moved forward from this point keeping these considerations in mind.

The second model we fit predict to Win Dividend is a Ridge Regression model. We chose to do Ridge next because it has a couple major benefits over ordinary least squares. The first benefit is that because Ridge Regression has a penalty term in its objective function, it tends to reduce over fitting, which is something we intend to do. The objective function for Ridge is as follows:

$$\text{minimize } (Y - B^T X)^2 + \lambda B^T B$$

Another benefit of using Ridge Regression is that it is guaranteed to produce a solution. Thus, with these benefits in mind, we decided to fit a Ridge Regression model. The solution for Ridge is:

$$\hat{B} = (\mathrm{X}^T X + \gamma Id)^{-1} X^T Y$$

Additionally, to perform Ridge Regression, we needed to find out the optimal α such that the regression model would find the best possible solution. In order to do this, we employed the Grid Search technique. Our intuition in using Grid Search is simple. Grid Search iterates through all possibilities in order to find the best parameters. Unlike Random Search, it covers the whole grid. Although Random Search may be faster in a lot of cases, we decided it was best to search through all possibilities. We discuss the results in "Model Results."

The next model we decided to fit is the Lasso model with an L2 regularizer. We decided to use an L2 regularizer because L2 regularizers are less sensitive to outliers, and based off our discoveries so far in predicting the Win Dividend, there are a few outliers that are dominating the total error. Thus, we believed this model could be useful in helping us to get a clearer view of how accurate our model is. Again, because we have also added a regularizer, this LASSO model tends to reduce over fitting. In this model, we also used the Grid Search method to find the optimal parameters to input for the model.

Finally, the last model we decided to fit is a Random Forest model. There are several reasons why we decided to use this model. First, Random Forest generally tends to reduce over fitting. As mentioned earlier, that is one of our goals. Furthermore, Random Forest is robust to outliers as it can essentially bin them. As we have discovered, this is important for our data set as there seems to be many outliers. The tricky part with implementing Random Forest as a model on our data is finding the right parameters. The quality of our results from this algorithm would depend on finding the optimal parameters. To find these parameters, as we did before, we employed the use of Grid Search. Although Grid Search will take longer than Random Search especially for the number of features we have, we wanted to make sure we got the right ones. Our results from this model are discussed in "Model Results" and we then examine which model produced the lowest error.

0.4.2 Win Probability

The second part of calculating the expected payoff is to calculate the win probability of every horse. As mentioned before, the win probability is the probability that a specific horse finishes in first place. As we did with Win Dividend, the first model we run on to predict Win Probability is the Least Squares model. Our intuition in choosing this model remains the same as before: we believe there is a linear relationship between the variables in the data set and the Win probability. This model will also serve as our default to which the other models can be compared. Some considerations we had going forward in choosing models included reducing over fitting as well as reducing the effect of outliers.

With this in mind, the second model we chose is a Ridge Regression model. As mentioned in Win Dividend, Ridge Regression reduces over fitting. As we also did with predicting Win Dividend, we used Grid Search to find the optimal parameters to use for Ridge Regression in Win Probability. We discuss our results in "Model Results" below.

For our third model, we decided to implement a logistic regression model. Our intuition behind choosing this model is simple. Because we are attempting to predict actual probabilities instead

of numbers, and logistic regression is a model known for providing probabilistic predictions, this model fits exactly what we need. Logistic regression will predict the actual probability of a specific event occurring, in our case, the probability a specific horse wins the race. For this model, in addition to MSE, we also calculate the Brier score, a score function that measures the accuracy of probabilistic predictions. Our results are discussed in "Model Results."

The fourth and last model we chose to fit on the Win Probability data set is the Random Forest. As we described in Win Dividend, the Random Forest tends to reduce over fitting as well as reduce the effect of outliers. Like before, we used Grid Search to find the best parameters for the algorithm. Our results are discussed in "Model Results" below.

0.5 Model Results

0.5.1 Error Metrics

To evaluate the performance of the models we trained to predict win probability, we used the mean squared error and a balanced Brier score (same as balanced mean squared error) of our models on the training set. The outcome for most rows in our dataset was a “loss” because the dataset came from many races of 14 horses where only one horse won in each of those races. Therefore, we had an unbalanced dataset and opted to use an error metric that equally weighted error among horses that won and that of horses that lost. The Brier score calculates the mean squared error between predicted probabilities and actual (0-1) outcomes. Since our predictions were in the range [0,1] (we set these limits for some models which gave predictions slightly outside of this range), we could use the Brier score as a metric to evaluate the performance of our win probability models. To adapt this metric to our unbalanced dataset, the balanced Brier score was calculated as such:

$MSE(pred, y_{won}) = \frac{1}{n}\sum(pred[i] - 1)^2$ over all i that had an actual outcome 1

$MSE(pred, y_{loss}) = \frac{1}{n}\sum(pred[i] - 0)^2$ over all i that had an actual outcome 0

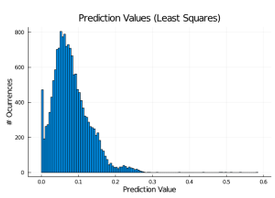
$BalancedBrierScore = 0.5 \times MSE(pred, y_{won}) + 0.5 \times MSE(pred, y_{loss})$

Model Type	Balanced Brier	MSE
OLS	0.40146	0.07056
Ridge	0.40144	0.07045
Logistic	0.42660	0.07326
Random Forest	0.40590	0.07127

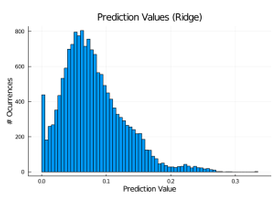
Table 1: Win Probability Models

Model Type	MSE	MAE
OLS	8869.6	61.05
Ridge	8800.4	60.93
LASSO	8794.5	60.91
Random Forest	6955.5	53.73

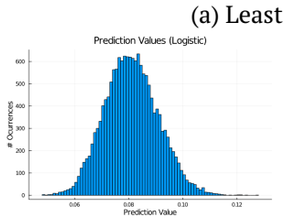
Table 2: Win Dividend Models



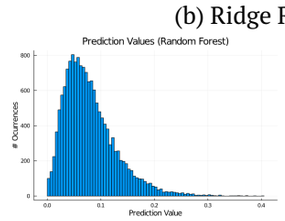
(a) Least Squares



(b) Ridge Regression

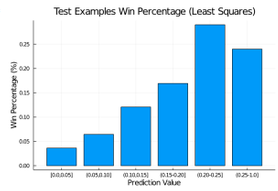


(c) Logistic Regression

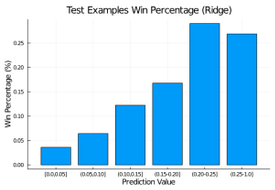


(d) Random Forest

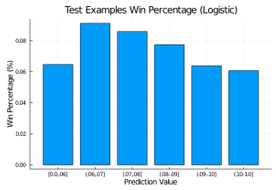
Figure 2: Win Probability Predictions



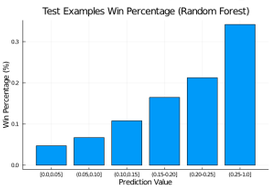
(a) Least Squares



(b) Ridge Regression



(c) Logistic Regression

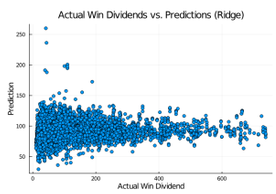


(d) Random Forest

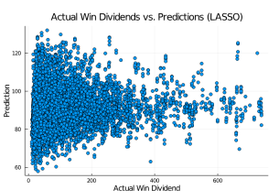
Figure 3: Predicted Win Probability vs Empirical Win Percentage

Our results, in table 1, show Ridge Regression model had the lowest MSE and balanced Brier score, with our Random Forest model and OLS model having close to as good metrics. To further evaluate the performance of our models, we created a histogram for each model of the empirical win percentage of examples in the test set that were assigned prediction values within different ranges by the model, shown in figure 3. We see that the Random Forest model appears to give more reliable probabilities to examples with higher win percentages than the other models, which we interpreted to mean that Random Forest may be more useful in betting strategies where one only wishes to bet on the horses with the best probability of winning.

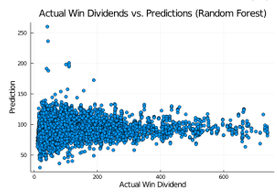
For evaluating the models we trained to predict win dividend, we used mean squared error (MSE) and mean absolute error (MAE), which would be less sensitive to outliers (which we knew would be present for win dividends). We decided that these would be the best measures of the accuracy of our models, which predict a continuous, real-valued label.



(a) OLS



(b) Ridge Regression



(c) Random Forest

Figure 4: Predicted vs Actual Win Dividends

Overall, we were not able to find a model that generalized win dividend very well. We determined this to be caused by a lack of features in our dataset that are correlated to higher win dividends. The dataset appears to mostly have data that would more likely be correlated to factors that affect a horse's performance in a race, rather than how payoffs are determined for a given race. The best we were able to achieve were models that mostly had predictions close to the mean of the actual labels (See Figure 4).

Of these models, Random Forest had significantly lower MSE and MAE than all three of the other models (See Table 2).

0.6 Conclusion

To test the usefulness of our models, we ran simulations of a betting strategy using the Random Forest model for predicting win probability and win dividend. We first calculated "expected values" of different bets by multiplying win dividend predictions by their corresponding win probability predictions from our models. Our goal was to see if using a strategy of betting on horses with the highest of these expected values could yield a profit. Our first simulation took a random sample of

2,000 examples from our testing set. Our two Random Forest models (trained on separate training data) made predictions of win probability and win dividend on all of these samples. We selected the 100 examples with the highest expected value (predicted win probability * predicted win dividend) and summed up the total expected value. According to the Hong Kong Jockey Club betting rules, a win bet has a unit cost of $10, so betting on 100 examples would have a total cost of $1,000, assuming we only bet once on each example. After calculating this expected profit, we calculated what our actual profit would be if we made these betting decisions, which we are able to calculate because our test set has the actual outcomes and dividends:

Run 1

Expected Profit: $1,148.90
Actual Profit: $2,342.50

Run 2

Expected Profit: $1,115.90
Actual Profit: $1,078.00

While this simulation yielded very promising results, we understand that it may take too much capital to bet on a 100 different races. Additionally, it may be impractical to calculate the 100 examples that have the highest expected payoff out of the 2000 examples under consideration. To simulate a smaller scale example, we bet on the 15 examples with the highest expected value out of 100 examples:

Expected Profit: $88.15
Actual Profit: -$131.00

We can see that there is a much greater risk when betting on a smaller number of races, and our models may only be able to reliably earn someone profits under a very large volume of betting and analysis.

Moreover, we cannot yet confidently say that our models can be used in production for our company's clients. Various factors about our model may not be widely applicable to horse race betting, as these rules and payoffs are specific to the Hong Kong Jockey Club. Many features in our dataset were very similar across all our examples, such as horse breed, track conditions, etc. Applying this model to races in other clubs or countries may yield greatly decreased performance. The results we were able to achieve using this dataset are promising in that we could earn large profits with high-volume betting. In the future, we see our models becoming more useful with the incorporation of new data sets from more diverse sources

0.6.1 Fairness - Weapon Math Destruction?

In this section, we will discuss the fairness of our model as well as whether it is a Weapon of Math Destruction. We will first discuss fairness. As we are predicting expected payoffs of betting on a specific horse in a specific horse race in Hong Kong, there are no "protected attributes" such as race or color that are in or data set. Furthermore, since none of these attributes are present, we can say that our model is unaware of these attributes. However, even though the model is unaware, considerations such as reduced accuracy, proxies, or allowable proxies do not apply in this case. Our model also satisfies demographic parity as the prediction is independent of any protected attributes. The fact that none of these protected attributes exist in our data set as we are dealing with horses instead of people leaves our model free of any unfairness. All horses also have equality of opportunity. There exists both individual fairness and counterfactual fairness.

Now we discuss whether or not our model is a weapon of math destruction. There is some potential for our predictive model to be a WMD. Our model predicts the expected payoff of betting on a horse in a specific race. However, this payoff depends a lot on the odds of that particular hose to win the race. And seeing as odds are largely influenced by the quantity of people betting on that specific horse and that specific race, our model in the hands of people could potentially change these expected payoffs. Because people will bet on the horses with higher expected payoffs, it will change the odds and therefore the payoffs. This can lead people to lose money even though they may believe they have a good change of winning money based off our model. Thus, there is potential for our predictive model to be classified as a WMD.