Brendan Leech and Colleen Caveney
CS451 Final Project Report

Bet On It: A Neural Nets Predictive Model for Hong Kong Horse Racing

Github Repository: https://github.com/bleech1/PredictingHorseRaces

Introduction:

For our final project, we wanted to investigate horse-racing results to see if we could classify how well a horse finished in a race based on pre-race data. In Hong Kong, horse racing is a very large industry, and horse betting has been shown to be profitable using machine learning.\textsuperscript{1} We attempted to replicate these results, albeit on a smaller scale, and see if we could create a model that would allow us to bet and consistently win money on horse racing.

Experimental Setup:

We used a dataset that we found on Kaggle titled “Horse Racing Dataset for Experts (Hong Kong).” Even though we are not experts on horse racing, we thought that this would be a fun and interesting challenge to tackle. This dataset contains five different tables, of which we used three. Two tables, the barrier and the results tables, contained race results that were the basis of our predictions. The barrier races are normally smaller races than those in the results table. We appended these two tables to each other, giving us just over 23,000 rows of data to use. We then joined this combined table with another table from the dataset that had information about the horses themselves. Combined, we were able to get features with information on both the horses and the race that the horse was participating
in. After randomizing the dataset, we split the data into train, cross-validation, and test sets using a 70%, 15%, 15% split.

Originally, we wanted to create a multiclass classifier to determine whether a horse placed first, second, third, or worse than third in each race. However, we modified our problem statement to determine whether a horse placed in the top three, also known as showing, or finished worse than third. To accomplish this, we created a binary classification neural network. To evaluate this network, we could not use accuracy since the results were skewed, with over 75% of our results in the “did not show” category. Therefore, we used precision, recall, and the F1 score to evaluate how our model was performing. To find out if our results were significant, we developed a betting strategy based on our model’s output and compared it to results from betting without the model. The strategy without the model was to bet $100 on the favorite in each race. With the model, we bet $100 on any horse that we predicted to show with confidence of at least 90%.

A large part of our time was spent making and testing features for our neural network. While we used some features directly from the dataset, like odds to win, we developed many of our features by hand. We included numerical representations of many of the string variables as well, such as the course and the horse’s sire. We developed features that calculated the number of times and percentage that each horse, trainer, and jockey won. We also added features to identify what horse was the favorite in its race and its average speed in its races. Finally, we added a feature that determined whether a horse was running in a longer race, over one mile, and over half of its previous four races were
shorter than a mile. This helped to show if a horse was not ready for a longer race. We developed these features according to our own intuition as well as evidence of success from previous predictive models from horse racing \cite{2}.

**Results:**

We are very pleased with the results that our neural network was able to produce. We achieved a precision of 0.27 (Figure 1) and a recall of almost 1 (Figure 2), giving us an F1 score of 0.43 (Figure 3). These numbers show that our neural network was classifying horses to show too often, as evidenced by our low precision but almost-perfect recall. In the case of betting on horses, betting horses to show often allows us to make more bets, albeit riskier with lower precision.

We were also thrilled by the betting results, where a bettor could use our model and have similar winnings to betting only on the favorites in races. After running our model for 50 runs on our test set, betting with our model generated an average of $3,314 more than betting on the favorite.

**Conclusions:**

Overall, this project was a success. While we did have a high false positive rate as shown by our low precision, our very high recall shows that we correctly predict showing for almost every horse that places in the top three. We accounted for the high false positive rate in our gambling strategy by betting on a horse to show only if the model was at least
90% confident in its prediction. This helped us to avoid betting on horses to show for many of the false positives.

While we are very happy with our results from this dataset, we believe that we could do even better with more data and additional features. More data would give us additional results for each horse and would also make some of our features, like a horse’s average speed, more robust. If we had more data, we would add features about horses’ opponents in a race, results from previous races where horses have competed in the same race before, and a rating for how well different horses race at different lengths.


Figure 1. Average precision from 4 epochs, over 50 runs of model training
Figure 2. Average recall from 4 epochs, over 50 runs of model training
Figure 3. Average F1-score from 4 epochs, over 50 runs of model training