Bet On Us

COMP 379 Final Paper

Dmitriy Dligach

December 16, 2017

*Horse Racing Prediction*

I. Introduction

A. For this project, our goal was to be able to predict if a horse would come in the top three positions, Win, Place, or Show, in a race. This information is useful when placing bets on horses at race tracks, which is a popular American pastime. To start, we wanted to find a dataset which mimicked a racing form, which is a document distributed on every race day that details each horse's statistics from previous races. These documents help betters know which horses to bet on, so we wanted to find something similar that had already been gathered in a spreadsheet. We found a github repository with a large dataset that had been used for a similar project, but we obviously wanted to take our own approach to analyzing it.

II. Dataset Description

A. The dataset contained information that is commonly found on racing forms, which included the features: post position, speed, horse/drive/trainer win percentage, horse/driver/trainer WPS percentage, horse/driver/trainer ROI, minimum races, previous break, days since last race, same track, same driver, last race result, last race win/place/show, last three races, and purse. Each value has been standardized to either -1, 0, or 1 by the creator of the dataset. This was fortunate for us, as this

made it easier to utilize by our models with minimal preprocessing. In order to turn this into a binary classification task, we wanted to predict whether a horse would WPS or not. So, we took the finishing position label and passed it through a lambda function. If the number was 1, 2, or 3, then it became a 1, and anything else became a 0.

III. Baseline Approach Description

A. We each chose to implement a different model to assess which would be the best performance on this dataset. The four of us implemented Support Vector Machine, K-Nearest Neighbors, and Decision Tree classifiers. From these options, we would then evaluate the models' performances and choose which one performed the best.

IV. Evaluation

A. Two different support vector classifiers were implemented. Because of the high-dimensionality of the dataset, Noel and John each chose two variables to analyze and use for their models. The support vector classifier creates a hyperplane from the training data; by comparing the test data against the hyperplane, it can be classified in one of the two categories. They wanted to determine which variables were more likely to succeed in placing bets. In the first classifier, the variables for same track and last race result were used. In the second classifier, horse ROI and last three races were used to train the model and make predictions. Although the two classifiers used different features, they both achieved around the same accuracy percentage of 73.

B. The K-Nearest Neighbors approach was taken because of the high-dimensionality of the dataset. K-Nearest Neighbors predicts the class of the testing data point based on its distance away from the neighboring training points. Because there were so many features, it may have been the best to be able to include all of them in the prediction. The standard implementation from the sci-kit learn package was used. Additionally, a grid search was performed to decide if Manhattan or Euclidean distance was the best, and to find the optimal value for neighbors. With Euclidean distance and nearest neighbors of 7, an accuracy of 71% was achieved.

C. Finally, Sebastian implemented a decision tree classifier. This model uses a decision tree to compare the feature values of a data point to the training data to make conclusions of the classification. In the tree created by the training data, the nodes represent class labels, and branches represent conjunctions of features that lead to those class labels. The classifier was implemented using the sci-kit learn library and tuned using a grid search for the best hyperparameters including gini to minimize misclassification and a max depth of 10. Ultimately, this model resulted in a 73% accuracy.

V. Discussion

A. This project was successful in finding which model and features worked best in predicting the winning horse of the race. Each model produced its own unique accuracy score which allowed us to determine which model and features were optimal. What we found interesting about our scores was the fact that they were all very similar. There was no model that outperformed the rest; all three models

outputted accuracy scores within a range of 70% to 74%. With different approaches and different algorithms, it was definitely a surprise to not have a single model surpass the rest.

## VI. Conclusion

A. In conclusion, the team was able to successfully explore and determine which of the following models tested would be the best as well as what features of the datasets were the most predictive. After using the same_track and last_race_res, we were confident we could predict seventy-three percent of the time which horse would place in the top 3. We are pleased with our results. If the project was to continue, a fascinating element would be to scrape data off the racing results websites and compare our predictions against the sites'.

## VII. Appendix

A. Bree Coffey worked on the KNN model as well as the written portions of the project. John O'Sullivan along with Noel Castillo implemented their own version of SVC with different features for comparison. Noel Castillo helped Bree with writing as well. Sebastian Kurpiel was responsible for working on the decision trees. He and John made the Google Docs presentation with the input of all the team members. We all met up in person throughout the span of the project, about weekly, to discuss our models/finds as well as delegate who would do what in terms of the final deliverables. The source code is available: https://github.com/johnosullivan/ml-horse-prediction