Beating the Odds: Machine Learning for Horse Racing

2019-11-27T15:27:56-05:00

Inspired by the story of Bill Benter, a gambler who developed a computer model that made him close to a billion dollars[@gambler] betting on horse races in the Hong Kong Jockey Club (HKJC), I set out to see if I could use machine learning to identify inefficiencies in horse racing wagering.

Data

The Hong Kong Jockey Club publishes all of the results for each race on their website. I created a script to scrape result cards from all of the historical available races. After running the script I was left with a dataset of 938 races spanning 14 months.

RACE 9 (207)

Class 3 - 1200M - (80-60)
ELGIN HANDICAP
HK$ 1,450,000

Going : GOOD
Course : TURF - "A" Course
Time : (23.48) (46.57) (1:09.91)
Sectional Time : 23.48 23.09 23.34

Multi Angle Race Replay

Plc.	Horse No.	Horse	Jockey	Trainer	Actual Wt.	Declar. Horse Wt.	Draw	LBW	Running Position	Finish Time	Win Odds
1	10	MERIDIAN GENIUS(C067)	C Schofield	A T Millard	119	1120	4	-	4 4 1	1:09.91	3.1
2	5	HIGHLY PROACTIVE(C458)	C Y Ho	K W Lui	124	1075	2	1/2	2 2 2	1:10.01	2.7
3	7	STARLIGHT(V273)	U Rispoli	C H Yip	120	1091	5	1-1/2	6 5 3	1:10.14	16
4	12	HAPPY WARRIOR(A313)	M Chadwick	J Moore	114	1098	7	1-1/2	8 7 4	1:10.17	14
5	3	PLANET STAR(A049)	N Callan	P O'Sullivan	129	1215	3	2	5 6 5	1:10.22	7.4
6	4	DIVINE UNICORN(C269)	L Hewitson	Y S Tsui	128	1214	6	3-1/4	9 9 6	1:10.44	85
7	6	THE RUNNER(D042)	S de Sousa	D J Whyte	121	1086	9	3-3/4	7 8 7	1:10.49	21
8	1	KASI FARASI(B385)	H W Lai	K H Ting	133	1062	1	3-3/4	3 3 8	1:10.51	13
9	9	E MASTER(A254)	R Bayliss	C Fownes	119	1136	10	4-3/4	11 10 9	1:10.66	19
10	11	FAIRY TWINS(V222)	M F Poon	W Y So	113	1171	12	5-1/4	12 12 10	1:10.76	105
11	8	AMBITIOUS HEART(V303)	A Sanna	D E Ferraris	119	1181	8	6-3/4	10 11 11	1:11.00	20
12	2	SUPER WISE(A100)	H T Mo	K L Man	129	1124	11	7	1 1 12	1:11.02	42

Result card from a HKJC race.

Feature Engineering

Going into this project, I had no industry knowledge about horse racing. Since not much information is provided with the race result cards, much work must

be done in engineering and selecting features in order to give a model more predictive power. Listed below are the features being used.

Draw: Which gate the horse starts in. This is randomly assigned before the race. Horses starting closer to the inside of the track (draw 1) general perform *slightly* better.

Horse Win Percent: Horse's win percent over the past 5 races.

Jockey Win Percent: Jockey's win percent over the past 5 races.

Trainer Win Percent: Trainer's win percent over the past 5 races.

Actual Weight: How much weight the horse is carrying (jockey + equipment).

Declared Weight: Weight of the horse.

Days Since Last Race: How many days it has been since the horse has last raced. A horse that had been injured in it's last race may have not raced recently.

Mean Beyer Speed Figure: Originally introduced in Andrew Beyer's *Picking Winners*[@beyer1994picking], the Beyer Speed Figure is system for rating a horses performance in a race that is comparable across different tracks, distances, and going (track conditions). This provides a way to compare horses that have not raced under the same circumstances. After reading Beyer's book, I implemented his rating system on my data. For this feature, I calculated the mean speed figure over the horse's past 5 race the race.

Last Figure: Speed figure of the last race the horse was in.

Best Figure at Distance: Best speed figure the horse has gotten at the distance of the current race.

Best Figure at Going: Best speed figure the horse has gotten at the track conditions of the current race.

Best Figure at Track: Best speed figure the horse has gotten at the track of the current race.

Engineering more features may yield better results; Benter's model[@benter2008computer] included many different types of features from many data sources.

Model

Before creating the model, it is important to understand the goal of the model. In order to not lose money at the race track, one must have an advantage over the gambling public. To do this we need a way of producing odds that are more accurate than public odds. For example, imagine the payout of horse is 5 to 1, and we have a model that indicates the horse's probability of winning is 0.2, or 4 to 1. Assuming our model is faithful, we would have an edge in this case, as we would be getting an expected return of $(0.2 * (5 + 1)) - 1 = 0.2$ or 20%. How do we create such a model?

Let's first assume the existence of a function R that provides a rating R_h of a horse h, given input features $x_h \in R^m$:

$$R_h = R(x_h)$$

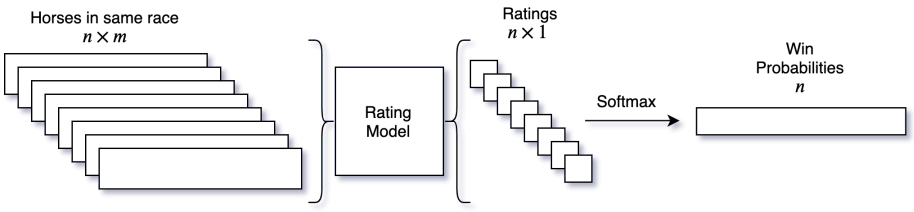
Assuming a horse with a higher rating has a higher probability of winning, we can compute the estimated probability of horse h winning, $\hat{p}_h$, given the ratings of all of the horses in the race:

$$\hat{p}_h = \frac{\exp(R_h)}{\sum_i \exp(R_i)}$$

Here we use the `softmax` function, as it's outputs will always sum to 1, and maintain the same order as the input.

Now that we know how we'll compute our probabilities, we must define our rating function R. For this we will use a neural network that takes an input vector of length m (where m is the number of features), and outputs a single scalar value. The structure of this network consists of two `Fully Connected` layers, each followed by a `ReLU`, `Batch Normalization` and `Dropout` layer. Lastly, there is a final fully connected layer to produce the single output.

Now we can visualize our model:



Training

We have defined our model, but how do we train it? For each race, let's call the winning horse w. If we had a perfect model, the predicted probability of w winning should be 1, that is $\hat{p}_w = 1$. We can encourage the model to approach this value by defining a loss function I'll call win-log-loss:

$$L(\hat{p}_w) = -\log(\hat{p}_w)$$

Win-log-loss will approaches 0 as the win-probability of the winner approaches 1, and approaches ∞ as the win-probability of the winner approaches 0. Now by minimizing win-log-loss via stochastic gradient descent, we can optimize the predictive ability of our model.

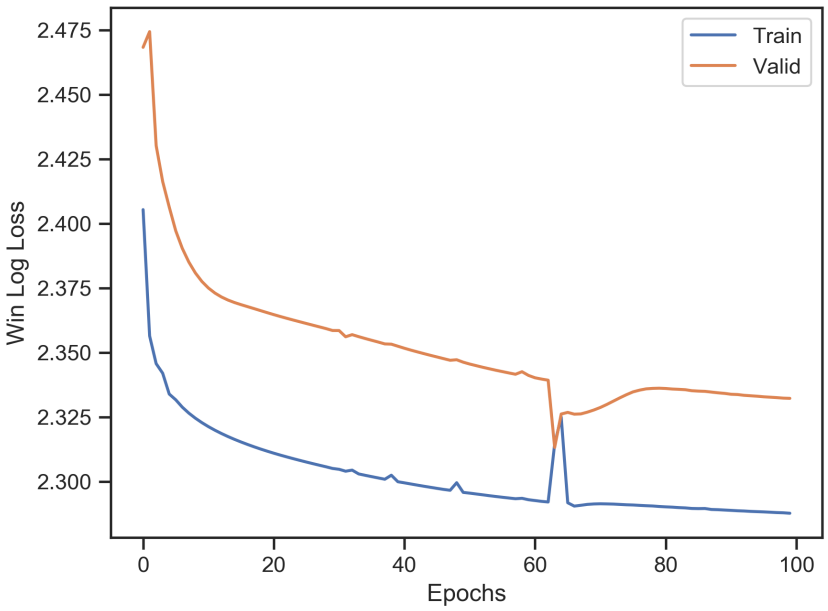
It is import to mention that this method is different than a binary classification. Since the ratings for each horse in a race are calculated using a shared rating network and then converted to probabilities with softmax, we simultaneously reward a high rating from the winner while penalizing high ratings from the losers. This technique is similar to a Siamese Neural Network, which is often used for facial recognition.

Betting

Now that we have predicted win probabilities for each horse in the race we must come up with a method of placing bets on horses. We can compute our own private odds for each horse using $1/\hat{p} - 1$. Now we *could* just bet on every horse whose odds exceed our private odds, but this may lead to betting on horses with a very low chance of winning. To prevent this, we will only bet on horses whose odds exceed our private odds, **and** whose odds are less then a certain threshold, which we will find the optimal value of over on our validation set.

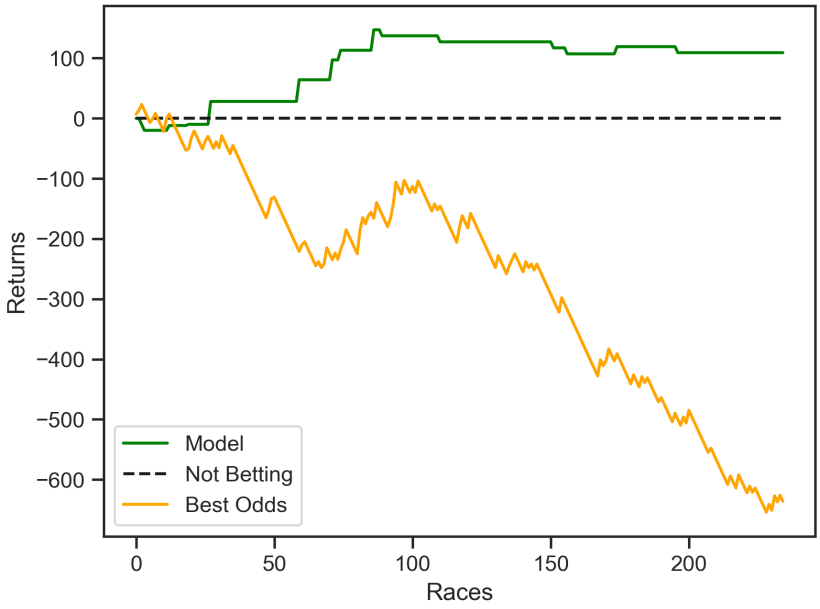
Results

We split the scraped race data chronologically into a training, validation, and test set, ensuring there would be no lookahead-bias. We then fit the horse-rating model to our training set, checking its generalization to the validation set:



After fitting the model, we find the optimal betting threshold on the validation

set, in this case `4.9`. We can now display our simulated results of betting 10 dollars on each horse that our model indicates. To compare, we also show the results of betting 10 dollars every race on the horse with the best odds, and one of the best strategies there is, not betting at all.



We can see that always betting on the horse with the best odds is a sure-fire way to lose all your money. While the model was profitable over both the training and validation sets, it is hard to say for sure how reliable it is because of the very low frequency of its betting.

Benter claimed that his era was a "Golden Age" for computer betting systems, where computer technology had only recently become affordable and powerful enough to implement such systems. Today, the betting market has likely become much more efficient with large numbers of computer handicappers and more detailed information available to the public. Nevertheless, developing a profitable model, especially with modern machine learning methods, may be a feasible task.

Thank you for reading! For any questions regarding this post or others, feel free to reach out on twitter: @teddykoker.