Horse Racing Prediction

Lau, KwanYuen* 20647191

Tang, HuiMin† 20636635

ABSTRACT

Horse racing became a fixture in Hong Kong in 1841 but enjoyment of this sport was originally confined to the city's elite. Today, a trip to one of the city's world-class tracks — Happy Valley on Hong Kong Island or Sha Tin in the New Territories — will confirm that it has very much gone mainstream since. During the whole racing season, hundreds of fans may grip their betting sheets to try their luck. However, picking the right horses for winning could be very difficult in practice. Thus, in our work, we decide to explore the possibility of providing insights about horse racing betting. To address this task, with the match data from 2007 to 2019, we propose a framework [1]. to predict the result of a given race, and also to give a betting strategy.

Index Terms: Hong Kong Horse Racing—Prediction—Betting Strategy

1 EXPLORATORY DATA ANALYSIS

For the data we use, involving 109,085 horses in 8,672 matches, each row represents a horse in a match. Specifically, a horse consists of 61 attributes, including:

- index
- rdate, rid, hid
- venue, track, going, course
- class, distance
- rfinishm, rm1, rm2, rm3, rm4, rm5, rm6
- horsenum
- jname, tname
- exweight, bardraw, gear
- rating, ratechg
- horseweight, horseweightchg
- besttime, age, priority
- lastsix, datediff
- rank
- runpos, p1, p2, p3, p4, p5, p6
- finishm, m1, m2, m3, m4, m5, m6
- windist, d1, d2, d3, d4, d5, d6
- pricemoney
- win_t5, win
- place_t5, place
- ind_win, ind_pla

Firstly, we should analyze the distribution of each attribute. However, we can notice that some attributes are highly correlated. Therefore, we drop part of the columns before this step, and finally get the output shown in Table 1.

Obviously, since **index** is the identifier of each row, we could simply neglect it. Then, we could divide the remaining attributes into three categories, i.e. Identifier, Characteristic and Ground Truth. Identifier, including **rdate**, **rid**, **hid** and **horsenum**, means attributes that we are not going to use as input of our model. Among them, **rdate**, **rid** and **hid** can uniquely define a horse in a match, and **horsenum** is the identifier of each horse. Ground Truth, including **rfinishm**, **rank**, **runpos**, **finishm**, **windist**, **win** and **place**, means attributes that are only available after matches. Lastly, Characteristic, including all the remaining attributes, will be used as input of our model.

2 IDEA AND RATIONALITY

For this part, we will not discuss any Identifier. With Table 1, we are able to decide which colums to be reserved, and also how to handle with null values.

Firstly, we can notice that, in some columns, most of the values are invalid. Considering the invalid values are hard to collect manually, we should drop these columns, including **gear**, **ratechg**, **horseweightchg**, **besttime**, **age**, **priority**, **lastsix**, **runpos** and **pricemoney**. Then, some columns are not suitable for being output of our model due to the lack of intuitiveness, and therefore we should drop them as well, including **rfinishm**, **windist**, **win** and **place**.

After dropping redundant columns, we need to fill the missing values. For **bardraw**, **exweight**, **horseweight**, **win_t5** and **place_t5**, we could naively fill them with their mean values. Nevertheless, in **finishm**, a missing value indicates that a horse didn't finish the match. Thus, we should fill it with a very large value, guaranteeing horses with small **finishm** performed better. Then, because **track**, **going**, **course**, **class**, **jname** and **tname** are with type of string, we are unable to fill them with a specific value. Therefore, we fill these columns with a crawler. These data can be collected on the official website of the Hong Kong Jockey Club [2].

3 FEATURE ENGINEERING

Further, we need to use domain knowledge of the raw data to create features that are easy to be processed by our model. In this part, we use data from the official website of the Hong Kong Jockey Club [3].

finishm In horse racing, different matches have different **distances**. Considering **finishm** could vary with different **distances**, we need to create a new feature called **velocity**, which is **finishm** devides **distances**, to eliminate this difference.

*e-mail: klauak@connect.ust.hk

†e-mail: htangak@connect.ust.hk

[1]All related code is available on https://github.com/HorseRacingPrediction/Model

[2]https://racing.hkjc.com/racing/Info/Meeting/ResultsAll/English/Local/

[3]https://racing.hkjc.com/racing/english/racing-info/racing_course.asp

class Horses can enter different races based on their handicap ratings. The original **class** is with type of string, which is difficult to be processed by our model. As a result, for a given horse, we convert its **class** into a numerical value based on the standard upper rating limit of its race class. Note that, since there are race classes that don't have standard upper rating limit, we use the average ratings of horses belonging to these classes instead. Additionally, there are some horses having invalid values in regard to **rating**, we fill these values with the corresponding values of **class**.

jname & tname **jname** and **tname** stand for the name of jockey and the name of trainer respectively. Initially, we intend to convert them into the historical average rank of jockey and the historical average rank of trainer respectively. However, we could find out that there are many jockey and trainer who have only appeared once. Consequently, the aforementioned scheme becomes arbitrary. To overcome this, we make an assumption that the appearance of a jockey or a trainer could reveal his capability. Moreover, because the appearances of different jockey and different trainer are various, we adopt the logarithm of their appearances to reduce the impact of this variance.

venue & course The rail of a match, which can be described by its home straight and its idth, depends on its **venue** and **course**. Thus, we could create two new features called **straight** and **width** to replace **venue** and **course**.

track & going **going** is the status of **track**, which is estimated by the penetrometer reading. Therefore, we could create a new feature called **humidity** to replace **track** and **going**.

normalization Lastly, after we convert all textual attributes into numerical attributes, we perform min-max normalization on each columns to scale to values to the range 0 to 1.

4 PREDICTION METHOD

To predict the result of a given race, we propose three different solutions. For each solution, we have implemented a residual neural network model and an ensemble tree model. To build an ensemble tree model, we use the LightGBM library. With LightGBM, we could easily build a random forest model, and apply cross validation to select appropriate hyperparameters. To build a residual neural network model, we apply dropout and batch normalization to prevent overfitting.

4.1 Regression

Because the final rank of a given horse depends on not only its performance but also its opponents' performance, predicting the rank of a given horse only based on its attributes seems to be improper. Hence, we want to seek a new criterion for it. One possible alternative is to predict the velocity of each horse instead, and calculate the rank of it by its velocity.

Since building an ensemble tree model is simple, we will not go deep into it. For the residual neural network, our model have two output, i.e. the predicted velocity and its corresponding confidence. To predict the velocity of a horse is straightforward, so our main task is to approximate the confidence. To obtain the confidence, we additionally penalize the difference between the ground truth and the predicted velocity of last iteration. However, if our model is poor at predicting the velocity, our training would be stochastic. We here introduce a new hyperparameter *weight*, which will gradually increase during training. We multipy our additional penalization term by *weight* to ensure that our additional penalization is negligible when our model is poor at predicting the velocity. Finally, we use the predicted velocity and its corresponding confidence to calculate the winning probability of a given horse in a match. The architecture of our network is shown as Figure 1.

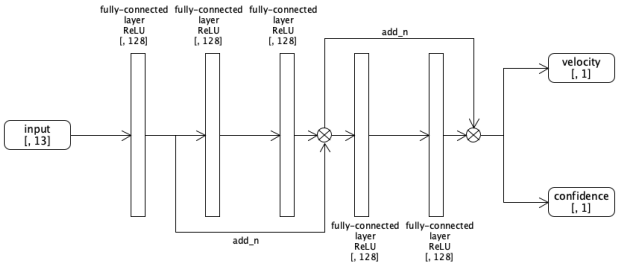

Figure 1: The architecture of regression network.

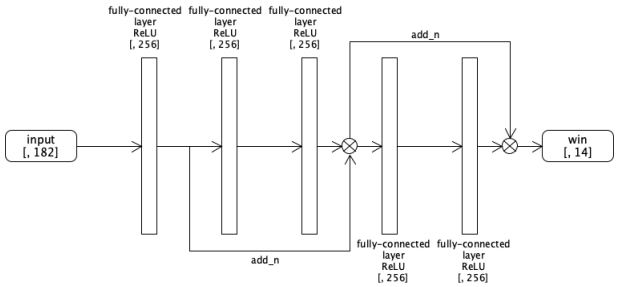

Figure 2: The architecture of classification network.

4.2 Classification

As we mentioned before, it is improper to predict the rank of a given horse only based on its attributes. Thus, we can group all horses involved in a match as a point and pass it into our model to obtain the winning probability of each horse in a match.

Here we also skip the details of the ensemble tree model. For this scheme, the main difficulty is that the numbers of horses involved in different matches are different. By analyzing the data, we can discover that there are at most 14 horses participating in a match. Hence, we can fix the number of attributes of a point to be 182 and fill those unnecessary attributes with 0. The architecture of our network is shown as Figure 2.

4.3 Naive Classification

Lastly, we have tried a naive scheme, which can help us examine our feature engineering rapidly. For details, we simply predict the rank of a given horse only based on its attributes. Then, we scale the output to ensure the sum of the winning probability of each horse in a match always equals 1. Note that there are 16 unique values with respect to **rank**, so we need to implement a 16-class classification. The architecture of our network is shown as Figure 3.

4.4 Post-processing

After obtaining the winning probability of each horse in a match, we also need to get the placing probability of each horse in a match. For regression model and classification model, we calculate the placing probability based on the winning probability. Specifically, we first estimate the 2nd-place probability and the 3rd-place probability, using the winning probability, and then sum up the winning probability, the 2nd-place probability and the 3rd-place probability. For naive classification model, we don't need to do any estimation, since we can get the 2nd-place probability and the 3rd-place probability from the output directly.

Finally, we need to determine a betting strategy. We simply choose a fixed ratio of bankroll and a merit threshold to get betting

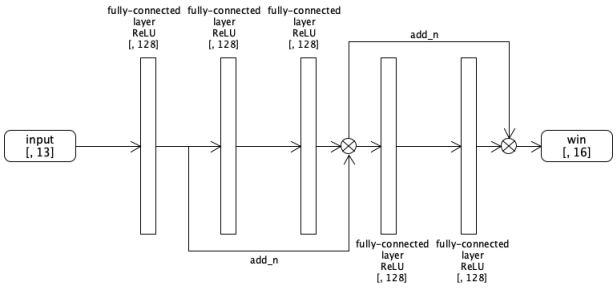

Figure 3: The architecture of naive classification network.

stake vectors of win and place.

5 Main Result

Since we only have leaderboards for five weeks, it is difficult to perform a complete analysis on our result. Although our model didn't perform well for the first few weeks, after improvement, our latest model could achieve relatively good performance on both the historical data and the back-testing data of each week. The result of back-testing based on the latest model is shown in Table 2.

6 Summary and Discussion

In practice, we could find out that although the ensemble tree models are easy to train, their performance is not ideal enough. Besides this, we have very limited approach to further improve them. According to the evalution of our framework, the residual neural networks have more potential than the ensemble tree models. Although the residual neural networks may perform well in many scenarios, it is hard for us to gain the robustness. Therefore, the main future goal of our work is to strike a balance between performance and robustness for the residual neural networks. Additionally, the naive scheme surprisingly outperforms the regression scheme and the classification, which are believed to be theoretically better. One possible explanation is that, we have dropped too many attributes, making it tough to train complicated models. Actually, we have also implemented other feature engineerings that are not currently used in our framework. We have attempted to convert **lastsix** into the average rank of the last six games; to compute the change of horses' weights based on the historical data; to compute the average stakes won of jockeys and the average stakes won of trainers. However, to accomplish these ideas, we may need to collect more extra data by crawler.

Table 1: Distribution of each attribute.

	index	rdate	rid	hid	venue	track	going	course
type	num	time	num	num	str	str	str	str
unique	109085	948	11	14	2	2	10	7
nan	0	0	0	0	0	5e-4	5e-4	0.14
mean	55543		5.3	6.9				
std	31848		2.8	3.8				

	class	distance	rfinishm	horsenum	jname	tname	exweight	bardraw
type	str	num	num	str	str	str	num	num
unique	22	9	2442	5507	211	174	32	16
nan	0.22	0	7e-3	0	1e-3	1e-3	0	5e-3
mean		1418	8441				122.3	6.8
std		276	1836				8.7	3.7

	gear	rating	ratechg	horseweight	horseweightchg	besttime	age	priority
type	str	num	num	num	num	str	num	str
unique	902	131	28	453	130	2937	9	12
nan	0.76	7e-3	0.70	3e-3	0.70	0.77	0.7	0.71
mean		70.0	-0.39	1106	0.53		5.05	
std		20.2	2.51	65.0	11.5		1.34	

	lastsix	rank	runpos	finishm	datediff	pricemoney	windist	win_t5
type	str	num	str	num	num	str	str	num
unique	28109	17	8979	5661	328	65	276	269
nan	0.70	0	0.71	5e-3	1e-3	0.46	3e-3	0.03
mean		6.83		8491	34.7			26.1
std		3.76		1904	38.4			26.9

	win	place_t5	place
type	num	num	num
unique	180	910	1297
nan	7e-3	0.10	0.10
mean	28.4	6.16	7.49
std	29.7	6.02	8.93

Table 2: Result of back-testing.

	Testing	Week1	Week2	week3	week4
AverageRMSEwin	0.2562	0.2458	0.2794	0.2572	0.2681
AverageRMSEpalce	0.3894	0.3804	0.4051	0.4045	0.3905
TotalProfit(Real odds)	0.0037	0.0007	0.0000	0.0023	0.0016
TotalProfit(Fair odds)	0.0051	0.0010	0.0000	0.0030	0.0021

	week5
AverageRMSEwin	0.2451
AverageRMSEpalce	0.3863
TotalProfit(Real odds)	0.0000
TotalProfit(Fair odds)	0.0000