STAT 8017

Data Mining Techniques

Who is the champion? HKJC horse racing prediction

Team Members

Du Shenghui	3036046787
Zhao Yibo	3036044258
Feng Bo	3036043541
Wang Jiashuo	3036043888
Tian Xiaodan	3036047353

Contents

Abstract

Hong Kong Jockey Club (HKJC) organizes and regulates horse racing events every week. Predicting the champion of the race has been a problem. However, the previous works only focus on pointwise ranking, and with no all-round pipelines for data mining. In this work, we not only scrapped 54436 racing records across 7 years, conducted feature engineering and selection, and built 5 ML models and 4 DL models that follow the traditions, but also innovatively apply pairwise training manner to tackle the problem, and developed python module for horse racing crawling, processing, and champion prediction. Further experiments and analysis proved our assumption is better over traditional ethics. We achieved at most 31.0% and 23.4% champion hit rate in validation and testing set. You can download our code in github: https://github.com/FanDazz/hkjc_race_ranking.

1. Introduction

1.1 Background

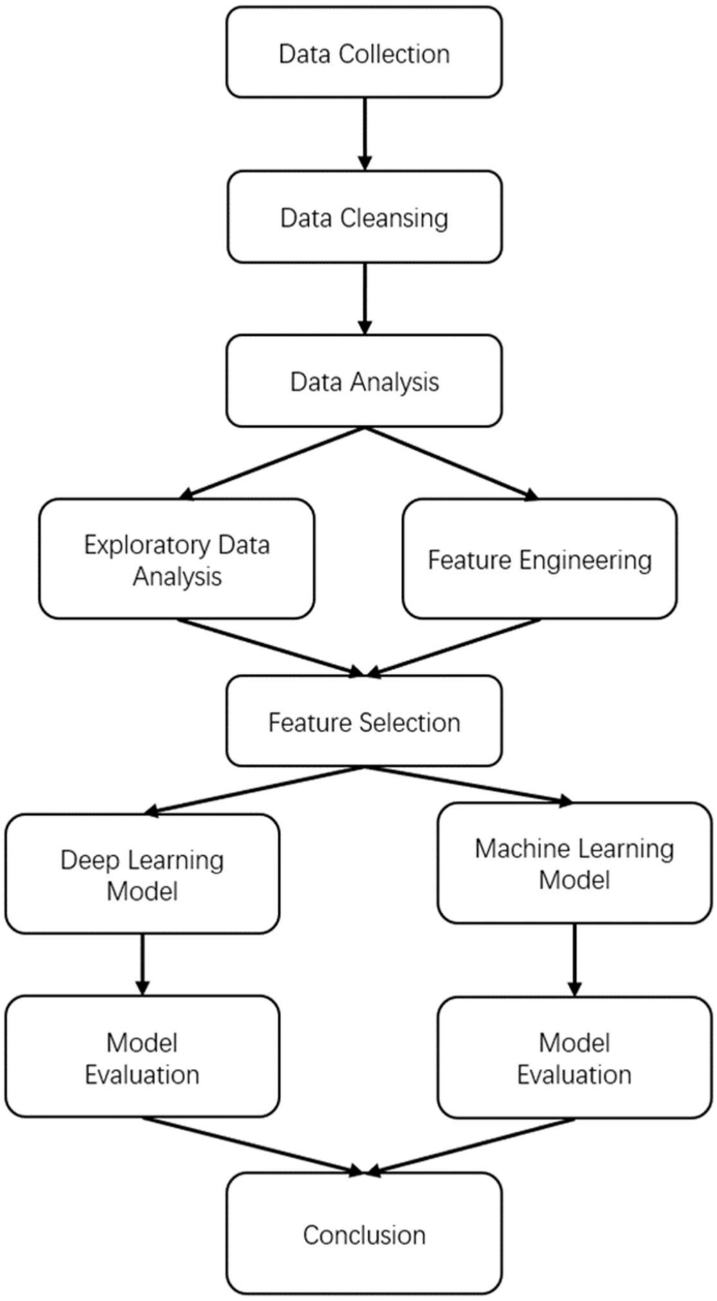
Horse racing is a popular sport and form of entertainment in Hong Kong. The Hong Kong Jockey Club (HKJC) organizes and regulates horse racing events throughout the year. As an important part of the HKJC's business, predicting the winning horse is crucial for both the club and the bettors. However, with many variables to consider, such as jockey, horse, train and past performance, accurately predicting the winning horse is a challenging task. To address this issue, we will use data mining techniques, including feature engineering, machine learning and deep learning methods, to predict the winning horse for the HKJC's upcoming horse racing session.

1.2 Analysis Task, Objective

The objective of this project is to predict the winning horse for the HKJC's upcoming horse racing session. To achieve this objective, we will perform the following analysis tasks:

1. **Data collection**: we will collect data from the HKJC's website, including information on the horse, jockey, trainer, racing distance, and so on.
2. **Data preprocessing**: we will clean the data and remove the outliers. We will also perform feature engineering to extract relevant features from the data.
3. **Feature selection**: we will explore the data of selecting a subset of relevant features, to reduce the dimensionality of the data and improve the accuracy and efficiency of the models. Model development: we will develop both machine learning and deep learning models to predict the winning horse based on the selected features.
4. **Model evaluation**: we will evaluate the performance of the model using various metrics. We will also compare the performance of our model with the betting odds.

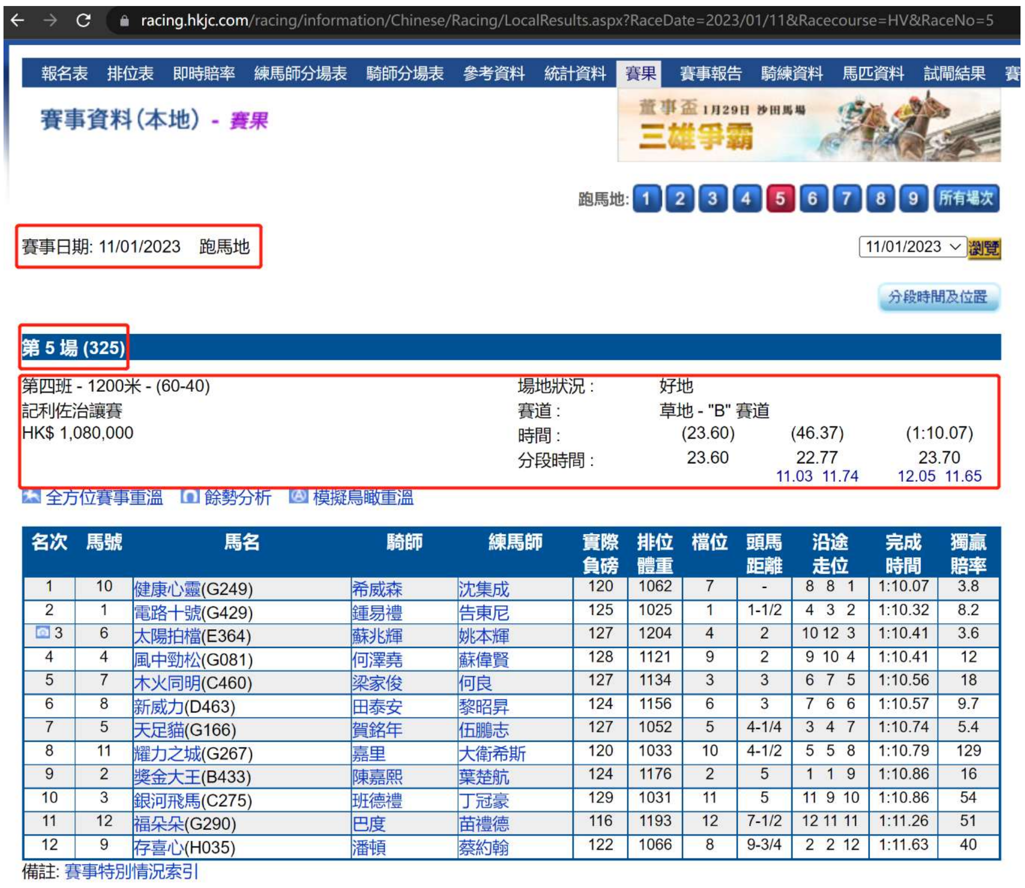
1.3 Workflow


Data Collection
Data Cleansing
Data Analysis
Exploratory Data Analysis
Feature Engineering
Feature Selection
Deep Learning Model
Machine Learning Model
Model Evaluation
Model Evaluation
Conclusion

2. Data Preparation

2.1 Scrapping, Cleansing Data from HKJC

All the data involved in our task were obtained from the HKJC's official website, and we used web scrapping techniques with selenium to obtain information about the races, riders, trainers, and horses. Here is a screenshot of HKJC website:


racing.hkjc.com/racing/information/Chinese/Racing/LocalResults.aspx?RaceDate=2023/01/11&Racecourse=HV&RaceNo=5
報名表 排位表 即時賠率 練馬師分場表 騎師分場表 參考資料 統計資料 賽果 賽事報告 騎練資料 馬匹資料 試閘結果 賽
董事盃 1月29日 沙田馬場 三雄爭霸
跑馬地: 1 2 3 4 5 6 7 8 9 所有場次
11/01/2023 瀏覽
分段時間及位置
全方位賽事重溫 餘勢分析 模擬鳥瞰重溫

賽事資料(本地) - 賽果

賽事日期: 11/01/2023 跑馬地

第 5 場 (325)

第四班 - 1200米 - (60-40)	場地狀況 :	好地		
記利佐治讓賽	賽道 :	草地 - "B" 賽道		
HK$ 1,080,000	時間 :	(23.60)	(46.37)	(1:10.07)
	分段時間 :	23.60	22.77 11.03 11.74	23.70 12.05 11.65

名次	馬號	馬名	騎師	練馬師	實際負磅	排位體重	檔位	頭馬距離	沿途走位	完成時間	獨贏賠率
1	10	健康心靈(G249)	希威森	沈集成	120	1062	7	-	8 8 1	1:10.07	3.8
2	1	電路十號(G429)	鍾易禮	告東尼	125	1025	1	1-1/2	4 3 2	1:10.32	8.2
3	6	太陽拍檔(E364)	蘇兆輝	姚本輝	127	1204	4	2	10 12 3	1:10.41	3.6
4	4	風中勁松(G081)	何澤堯	蘇偉賢	128	1121	9	2	9 10 4	1:10.41	12
5	7	木火同明(C460)	梁家俊	何良	127	1134	3	3	6 7 5	1:10.56	18
6	8	新威力(D463)	田泰安	黎昭昇	124	1156	6	3	7 6 6	1:10.57	9.7
7	5	天足貓(G166)	賀銘年	伍鵬志	127	1052	5	4-1/4	3 4 7	1:10.74	5.4
8	11	耀力之城(G267)	嘉里	大衛希斯	120	1033	10	4-1/2	5 5 8	1:10.79	129
9	2	獎金大王(B433)	陳嘉熙	葉楚航	124	1176	2	5	1 1 9	1:10.86	16
10	3	銀河飛馬(C275)	班德禮	丁冠豪	129	1031	11	5	11 9 10	1:10.86	54
11	12	福朵朵(G290)	巴度	苗禮德	116	1193	12	7-1/2	12 11 11	1:11.26	51
12	9	存喜心(H035)	潘頓	蔡約翰	122	1066	8	9-3/4	2 2 12	1:11.63	40

備註: 賽事特別情況索引

Fig 2.1.1 Screenshot of HKJC Website of Horse Racing Performance

With our developed available in *.horse.scrapper*, whose underlying logic may not be our focus in this report. If you are interested in how we go over all races in each racing day, and scrap data inside each page, you are warmly welcomed to have a look. In short, **we scrapped 4552 races from April.22nd, 2015 to Jan.18th, 2023, with 54436 records in 21 columns**, and **scrapped supplementary information on jockeys, trainers and horses from HKJC**.

Next, we are ready to cleaned these records. Since we mainly use horse racing performance data, which has a detailed and accurate record in HKJC, we do not need to deal with the missing values of the data in question. Hence, we should only focus on invalid participants:

1. **Remove horses that are withdraw or be suspended** from the race, out of broken rules, over excited, got injured, or other reasons, around 0.8% of the records.
2. **Remove races with few participants**, which may out of unsuccessful requirements, banned

websites, etc, affected nearly 1% or the data.

2.2 Data Overview

Before performing feature engineering and selection, we analyzed the raw data just scrapped and identified significances of all the features:

FEATURE NAME	SIGNIFICANCE
Race Data	The exact day of a race.
Race No	The session order of the race.
Race Course	The competition course of the race
Distance	The distance of a race session.
Field Going	The status of the competition course.
Race Name	The name of the race.
Course Type	The course type of a race session.
Race Money	The race money of a race session.
Pla.	The ranking of a race session.
Horse No.	The index of the horse in a race session.
Jockey	The name of the racing Jockey in this session.
Trainer	The name of the trainer of racing horse in this session.
Act. Wt.	The extra weights on a horse to balance their self-weight.
Declare. Horse. Wt.	The self-weight of a horse.
Dr.	The starting barrier of a horse in a race session.
LBW	The distance to the leader horse at the end of a race session.
Running Position	The rankings of a horse at key points in a race session
Finish Time(min)	The minute(s) portion of the finish time
Finish Time(sec)	The second(s) portion of the finish time
Win odds	The winning odds of the horse in the race session

Table 2.2.1 Feature Introduction

To gain a better understanding of the data and identify potential patterns, we further explored the data by visualizing potential available features:

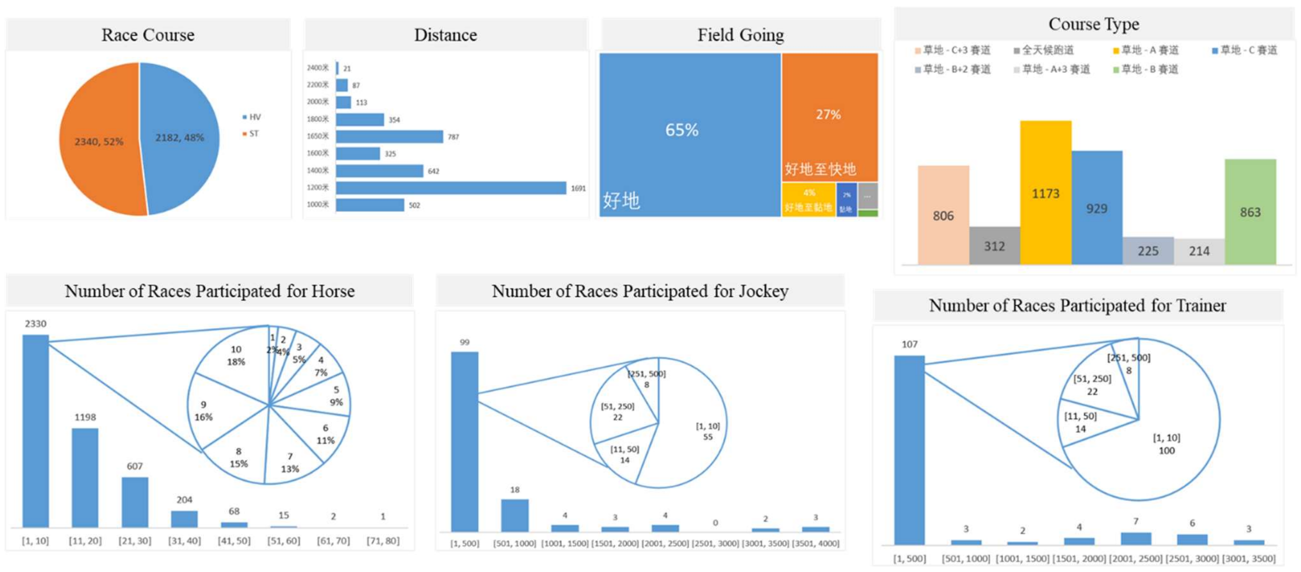


Fig 2.2.1 Visualization of Raw Features

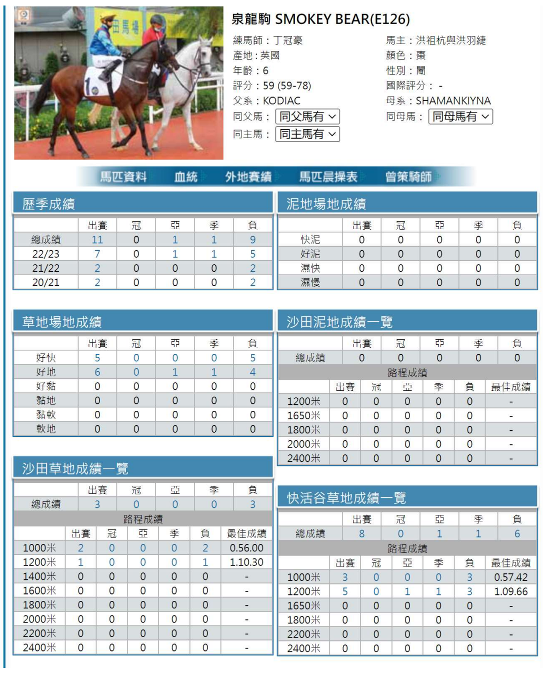
Based on the above content, we can observe that:

1. The distribution of Race Courses is fairly even.
2. Most races are between 1000-1800 meters.
3. Field Going is mainly concentrated in ‘好地’ and ‘好地至快地’.
4. Course Type is mainly concentrated in ‘草地-A 赛道’, ‘草地-B 赛道’, ‘草地-C 赛道’, ‘草地-C+3 赛道’.
5. The number of times horses participate in races is concentrated within 40 times.
6. Most jockeys participate in races less than 10 times, but some jockeys participate in races more than 3000 times.
7. Most trainers participate in races less than 10 times, but some trainers participate in races more than 3000 times."

3. Feature Engineering

3.1 Motivation, Framework

Racing paper would be one of the most popular channels for horse-racing fans to acquire information, and all-round statistics, analysis on participants in each game, since it provides a wealth of information about the rider, horse, and trainer. Within each racing paper, statistics would be made regarding participants' historical performances, and specialists' rating on them, or the combination of them to illustrate how well they are under collaboration, e.g. combination of horse and jockey. For example, in one racing paper's website, records of historical performance are detailed illustrated, which would be helpful for making a guess on champion:



泉龍駒 SMOKEY BEAR(E126)

練馬師：丁冠豪	馬主：洪祖杭與洪羽緹
產地：英國	顏色：棗
年齡：6	性別：閹
評分：59 (59-78)	國際評分： -
父系：KODIAC	母系：SHAMANKIYNA
同父馬： 同父馬有	同母馬： 同母馬有
同主馬： 同主馬有	

歷季成績

	出賽	冠	亞	季	負
總成績	11	0	1	1	9
22/23	7	0	1	1	5
21/22	2	0	0	0	2
20/21	2	0	0	0	2

泥地場地成績

	出賽	冠	亞	季	負
快泥	0	0	0	0	0
好泥	0	0	0	0	0
濕快	0	0	0	0	0
濕慢	0	0	0	0	0

草地場地成績

	出賽	冠	亞	季	負
好快	5	0	0	0	5
好地	6	0	1	1	4
好黏	0	0	0	0	0
黏地	0	0	0	0	0
黏軟	0	0	0	0	0
軟地	0	0	0	0	0

沙田泥地成績一覽

	出賽	冠	亞	季	負
總成績	0	0	0	0	0

路程成績

	出賽	冠	亞	季	負	最佳成績
1200米	0	0	0	0	0	-
1650米	0	0	0	0	0	-
1800米	0	0	0	0	0	-
2000米	0	0	0	0	0	-
2400米	0	0	0	0	0	-

沙田草地成績一覽

	出賽	冠	亞	季	負
總成績	3	0	0	0	3

路程成績

	出賽	冠	亞	季	負	最佳成績
1000米	2	0	0	0	2	0.56.00
1200米	1	0	0	0	1	1.10.30
1400米	0	0	0	0	0	-
1600米	0	0	0	0	0	-
1800米	0	0	0	0	0	-
2000米	0	0	0	0	0	-
2200米	0	0	0	0	0	-
2400米	0	0	0	0	0	-

快活谷草地成績一覽

	出賽	冠	亞	季	負
總成績	8	0	1	1	6

路程成績

	出賽	冠	亞	季	負	最佳成績
1000米	3	0	0	0	3	0.57.42
1200米	5	0	1	1	3	1.09.66
1650米	0	0	0	0	0	-
1800米	0	0	0	0	0	-
2200米	0	0	0	0	0	-
2400米	0	0	0	0	0	-

Fig 3.1.1 Screenshot of Racing Paper's Website

Motivated by these works, we need to conduct feature engineering to generate more features, whose framework can be illustrated in **Fig 3.1.2**.

Our core idea for creating new features was to follow the diagram above, add statistical features, and made our carefully tailored supplementary features:

Statistics of each entry, such as horse, jockey, trainer with all-round information, covering its historical-wide, long-, mid-, and short-term performance. For example, statistics for each race depending on the time span, wining rate, top4 rate, lifetime, etc, in different granularity in year, month, and days.

Detailed information is available in the **Table 3.1.1**. Please be noted that since we have constructed new features with the same logic for all entries, only mian ideas will be covered.

Besides, ranking scores like the ones provided by specialist will also be considered. Since ratings by specialists are obscure, and acquiring these scores is hard, not to mention the missing of ground truth behind these scores. Hence, we would construct ELO scores, whose details would be introduced in **3.2**.

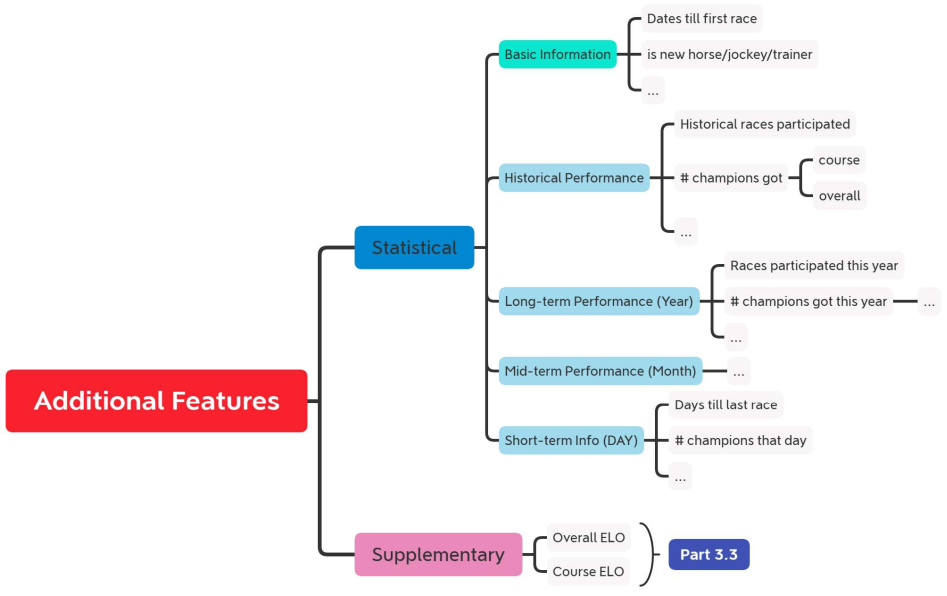


Fig 3.1.2 Framework of feature engineering

Performance Features by statistics	Total number of races in history to date/in last year/in last month/ in last 5 races
	Length of career (in days)
	The best place in the history
	Total number of winning in history to date/in last year/in last month/ in last 5 races
	Winning rate to date/in last year/in last month/ in last 5 races
	Total number of top 3 in history to date/in last year/in last month/ in last 5 races
	Top 3 rate to date/in last year/in last month/ in last 5 races
	Total number of top 4 in history to date/in last year/in last month/ in last 5 races
	Top 4 rate to date/in last year/in last month/ in last 5 races

Table 3.1.1 Performance Features

3.2 Engineering Feature: ELO

The raw data contains some non-numeric features that are essential for understanding and predicting outcomes. Traditional numerical methods struggle to incorporate these non-numeric features, leading to incomplete or inaccurate models. However, the use of ELO ratings provides a powerful solution to this problem: ELO incorporates them into its calculations, providing a more comprehensive and accurate assessment of the relative strengths of different participants. This approach can lead to more robust and interpretable models, which in turn can improve decision-making and drive better outcomes.

We are using two types of ELO ratings combining non-numeric information on three kinds of entities: horse, jockey and trainer.

The first type is individual-entity ELO. This ELO reflects an entity's performance relative to its competitors in its historical race session, the formula is:

$$ELO_{(i),r} = ELO_{(i),r-1} + K * (\frac{\sum_{j=1}^{N} FT_{j,r}}{N} - FT_{i,r})$$

i: the index of the entity.

r: the race session sequence index.

K: the adjustment step size

$FT_{j,r}$: the finish time of entity j in race session r

N: the total number of participants in this race session

The second type is course-combined ELO. This ELO reflects an entity's performance relative to their competitors on different types of racing course. The formula is:

$$ELO_{(i,d,t),r} = ELO_{(i,d,t),r-1} + K * (\frac{\sum_{j=1}^{N} FT_{(j,d,t),r}}{N} - FT_{(i,d,t),r})$$

i: the index of the entity.

r: the race session sequence index.

K: the adjustment step size

$FT_{j,r}$: the finish time of entity j in race session r

N: the total number of participants in this race session

d: the distance of the race session

t: the course type of the race session

3.3 Exploratory Data Analysis & Insights on Saturated Features

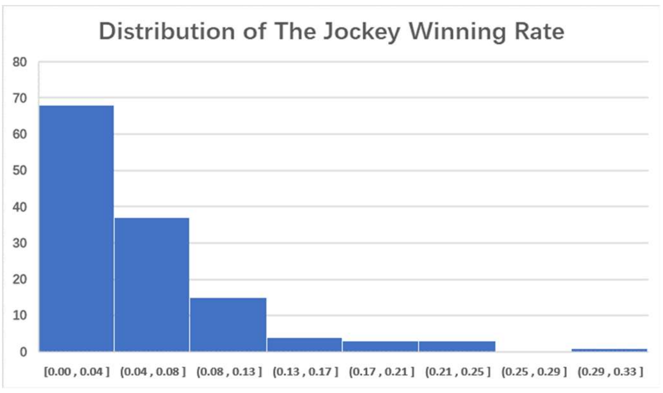


Fig 3.3.1: Distribution of the jockey winning rate.

We have a total 131 jockeys, and there are 51 jockeys who have never won a race. The odds of winning for the remaining jockeys are shown in the following figure, where we can see that the odds of winning for the HKJC's ace jockeys are around 0.17 to 0.21, with some jockeys with odds of 0.25 or more having competed in fewer races. The average jockey's winning rate distribution is between 0 and 0.17.

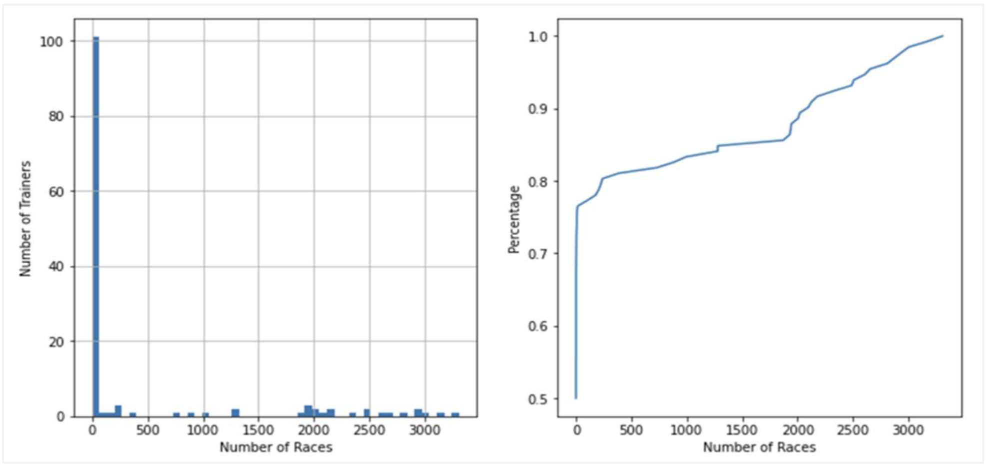


Fig 3.3.2: Number of races participated by trainers, in p.d.f. and c.d.f

From the above figure we can find that the total number of trainers we have in our data is 129, and trainers at most attend for 3307 games in total. The position of trainer is somewhat unusual because generally horses competing in Hong Kong are sent to specific barns for training and the number of qualified barns that we have in Hong Kong is limited, but there is no shortage of horses competing in Hong Kong from other countries or regions, so that would make our data have many trainers with less than 20 races, and the number of it reached 98. Half of the trainers attend only 1 race within the past 15 years, but 23% have attended more than 100 races.

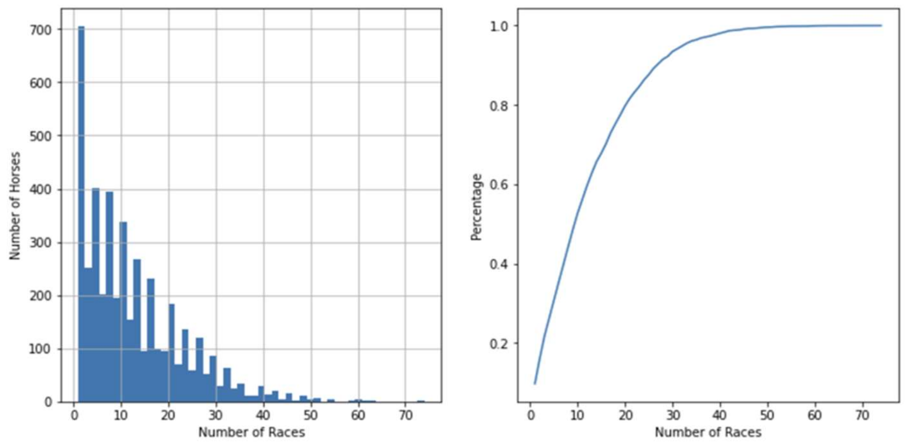


Fig 3.3.2: Number of races participated by horses, in p.d.f. and c.d.f

Based on the above figure, we can find that 50 percent of the horses attend less than 10 races within the past 15 years. Horses take at most 74 races, with only 3.9 percent having attended more than 35 races.

In addition, we will show here the distribution and characteristics of our new features.

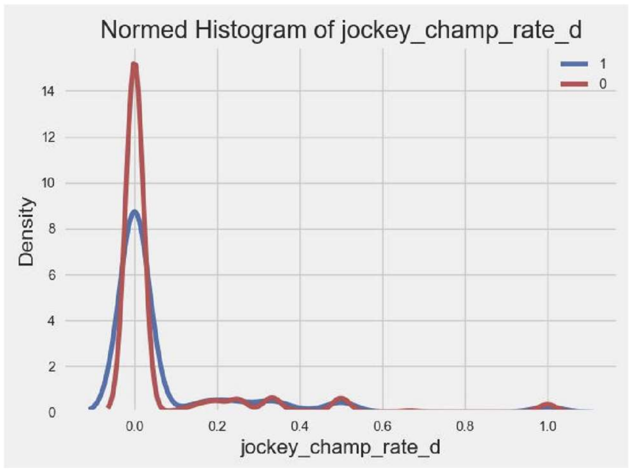


Fig 3.3.4: Probability density curve of champion and jockey_champ_rate_d

The first feature is jockey_champ_rate_d which represents the rate of specific jockey who have won the championship on the current race day. Following figure is normed histogram. 1 represents the jockey who gets the champion and 0 represents the jockey who does not get the champion. And from the figure we can find that even if a jockey has not won on the day of this race, he/she still has a chance to win the next race. And those who have not won before have a better chance of winning. This corresponds to the actual situation where each barn uses the field strategy for each day of the race.

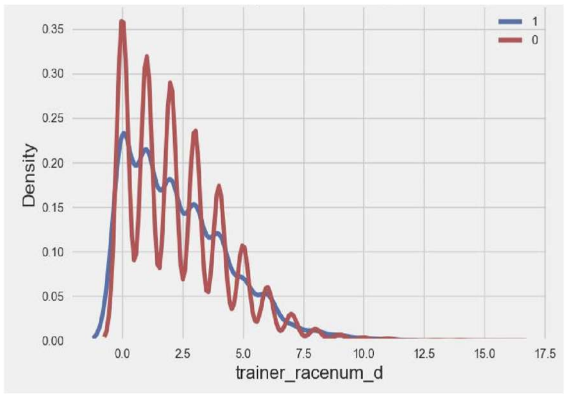


Fig 3.3.5: Probability density curve of champion and trainer_racenum_d

The second feature is trainer_racenum_d which represents the number of specific trainers who have won the championship on the current race day. And from the above figure we can find that as the number of races increases, trainers are less likely to win and most trainers can offer less than 8 horses per racing day.

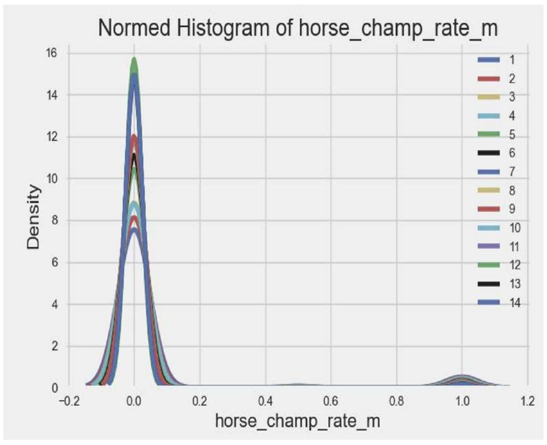
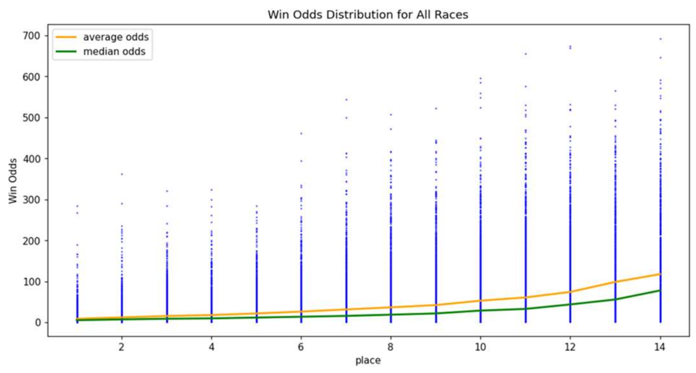
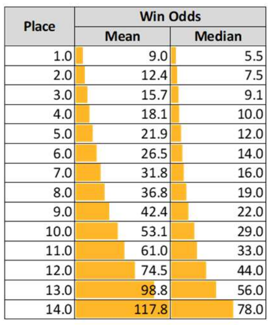


Fig 3.3.6: Probability density curve of place and horse_champ_rate_m

The third feature is horse_champ_rate_m which represents the rate of horses winning each month and the number of places they race to get. And from the above figure we can find that even if a horse has a probability of winning in the past month of 1, they still have a chance of finishing very low on the list. Conversely, even if they have a low probability of winning in the past, they still have a good chance of finishing well.





Place	Win Odds	
	Mean	Median
1.0	9.0	5.5
2.0	12.4	7.5
3.0	15.7	9.1
4.0	18.1	10.0
5.0	21.9	12.0
6.0	26.5	14.0
7.0	31.8	16.0
8.0	36.8	19.0
9.0	42.4	22.0
10.0	53.1	29.0
11.0	61.0	33.0
12.0	74.5	44.0
13.0	98.8	56.0
14.0	117.8	78.0

Fig 3.3.7: Distribution of Winning Odds

According to the observation of the distribution diagram of place and win odds, it can be known that the variable distribution is basically in line with expectations, and also has a certain linear relationship. The higher the place, the lower the win odds, and vice versa. This is also in line with most of the conditions in the race. The favorite horses generally have lower win odds. The mean and median of win odds in different places can also prove the linear relationship between the two variables.

3.4 Missing Value and Outlier

There is no missing data, so no processing is required. On the other hand, as for outliers, we analyzed each feature individually and found the following pattern on Dr.

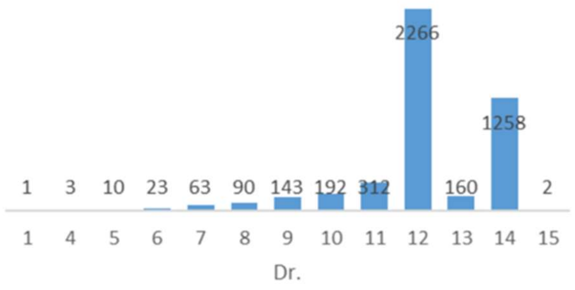


Fig 3.4.1: Total number of race sessions with different Dr.

From the figures above, we can find that the Dr. distributes mainly between 9 and 14. Moreover, it is also too weird to find that some race sessions only have 1 or 4 barriers, which means only 1 or 4 horses were competing for the champion in a session. We identify these kinds of records as outliers, and we would drop them. And to guarantee the statistical significance of these features, we only keep the records with more than 100 sessions, that is the Dr. between 9 and 14.

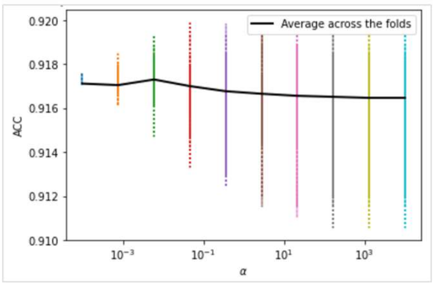
3.5 Feature Selection with L1-normed Logistic Model

With much work on feature engineering, comprehensive descriptions for all entries within each race is drawn, and it is indispensable to conduct feature selection to step ahead. That is, filtering out the most important features from 100+ features before further modelling, otherwise, models might be too heavy to be either interpretated or easily trained.

In align with the problem of champion prediction, we propose an assumption that each racing entry is distributed independently with each other, and their probability to win is only affected by his features, i.e. $\pi_i = \pi(x_i) = E(y_i|x_i)$, where x_i stans for entry i's feature, y_i is its wining outcome, and π_i is the expected probability. To make things easier, we further laid a linear relationship between all features and response, and removed categorical features.

Given these assumptions, we applied 20-fold cross validation wih Logistic Regression with L1 penalty to see the performance of Logistic Regression given different size of features. As shown in **Fig 3.5.1** and **Fig 3.5.2**, accuracy culminates when L1 penalty=1/0.0008, and the standard deviation across each fold is relatively low, indicating reliable, and stable predictions given the 22 features. Besides, since the label is distributed imbalancely, we need to see the average precision (AP) of our prediction. Obviously, adding more features over 57 achieve better performance w.r.t. training AP, the validation AP is nearly the same as 22. Moreover, the training time of high penalty coefficient would be hard to tolerate with over 100 seconds for merely logistic model.

Hence, given the reliability of prediction, decent validation AP, fast training time, and light number of features, we L1 penalty=1/0.0008, when the number of rest features=22 would be the most suitable choice in our condition.



L1 Penalty (Inverse)	Rest # Features	Training Time (seconds)	Training Accuracies		Average Precision (%)	
			Mean	Standard Deviation	Training	Validation
0.0001	0	5.14	91.71%	0.0002	8.28%	8.46%
0.0008	2	5.06	91.70%	0.0006	24.51%	23.26%
0.0060	22	5.37	**91.73%**	0.0011	25.29%	26.48%
0.0464	57	8.85	91.70%	0.0016	27.49%	26.70%
0.3594	94	108.88	91.68%	0.0021	28.30%	27.97%
2.7826	107	136.63	91.67%	0.0023	28.69%	27.12%
21.5443	108	133.20	91.66%	0.0024	28.74%	26.91%
166.8101	108	150.37	91.65%	0.0025	28.72%	27.33%
1291.5497	108	133.35	91.65%	0.0025	28.66%	27.12%
10000.0000	109	134.02	91.65%	0.0025	28.66%	27.33%

Fig 3.5.1: Accuracy of LR under different L1 penalty; **Fig 3.5.1** Performance of LR under different L1 penalty

Then, we again fit our Logistic model with L1 to acquire the most important 22 features, and incorporate the set into our dataset classes' attribute (check details in *horse.data.load_data.DataSet*).

Afterwards, the downstream models can access the most important features easily. Although features are selected linearly, but we think there lies memorable relations against the response, and further high-order relations can be extracted from these features. Hence, they would be also suitable for other non-linear models like decision tree, XGBoost, and MLPs.

More details on features, their corresponding coefficients, and interpretation is available in 5.3 KQ3.

3.6 Handling Categorical Features

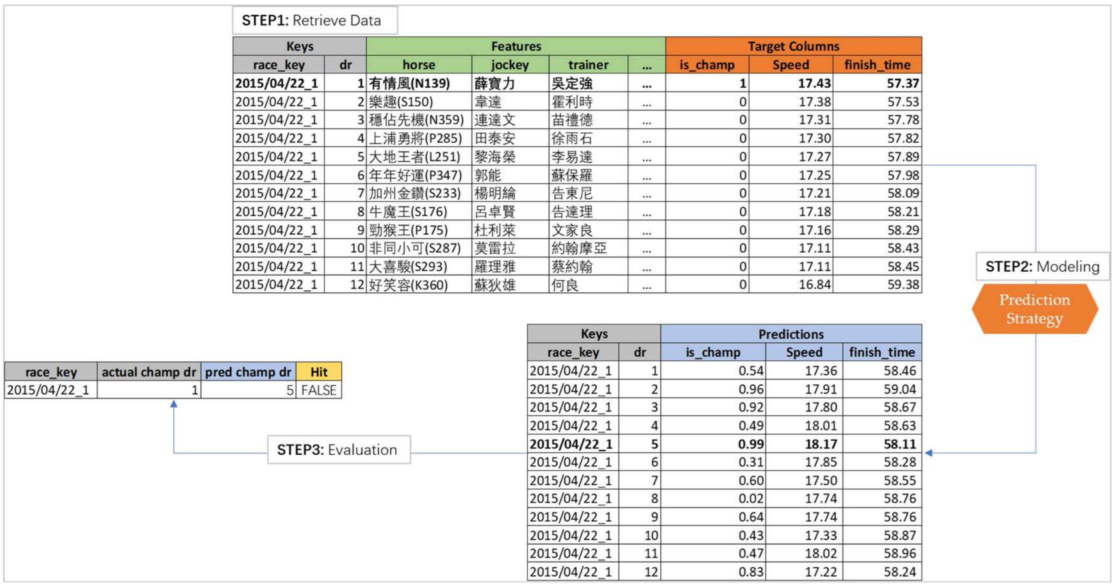
The very first trial is to encode each variable with one-hot vector. Such trail is also suitable for all situations, either ML or DL. However, such encoding is not time-variant, and is universal to each entry across time. There comes information encoding, i.e. depict entries with features, such as historical winning rate, racing times, etc, as what we have done in the above parts, which is changeable across the time, and do perceive information of the entry to some extent.

Inspired by recommendation system, and NLP, we can also encode each entry with embeddings. That is, represent each entry, e.g. horse, trainer, or jockey, with a k-dimensional vector. The representation would be successful as long as k<<cardinality of the entry. Since torch provides more customization over scikit-learn, we can implement it in the DL-based models.

4. Models for Champion Prediction

4.1 Problem settings

In order to predict champion, one can make predictions on certain criteria, e.g. speed, racing time, wining probabilities, scores, etc, and ranking each participant accordingly to retrieve the result. From Fig 4.1.1, one can see the detailed steps:



Keys		Features				Target Columns		
race_key	dr	horse	jockey	trainer	...	is_champ	Speed	finish_time
2015/04/22_1	**1**	**有情風(N139)**	**薛寶力**	**吳定強**	**...**	**1**	**17.43**	**57.37**
2015/04/22_1	2	樂趣(S150)	韋達	霍利時	...	0	17.38	57.53
2015/04/22_1	3	穩佔先機(N359)	連達文	苗禮德	...	0	17.31	57.78
2015/04/22_1	4	上浦勇將(P285)	田泰安	徐雨石	...	0	17.30	57.82
2015/04/22_1	5	大地王者(L251)	黎海榮	李易達	...	0	17.27	57.89
2015/04/22_1	6	年年好運(P347)	郭能	蘇保羅	...	0	17.25	57.98
2015/04/22_1	7	加州金鑽(S233)	楊明綸	告東尼	...	0	17.21	58.09
2015/04/22_1	8	牛魔王(S176)	呂卓賢	告達理	...	0	17.18	58.21
2015/04/22_1	9	勁猴王(P175)	杜利萊	文家良	...	0	17.16	58.29
2015/04/22_1	10	非同小可(S287)	莫雷拉	約翰摩亞	...	0	17.11	58.43
2015/04/22_1	11	大喜駿(S293)	羅理雅	蔡約翰	...	0	17.11	58.45
2015/04/22_1	12	好笑容(K360)	蘇狄雄	何良	...	0	16.84	59.38

Keys		Predictions		
race_key	dr	is_champ	Speed	finish_time
2015/04/22_1	1	0.54	17.36	58.46
2015/04/22_1	2	0.96	17.91	59.04
2015/04/22_1	3	0.92	17.80	58.67
2015/04/22_1	4	0.49	18.01	58.63
2015/04/22_1	**5**	**0.99**	**18.17**	**58.11**
2015/04/22_1	6	0.31	17.85	58.28
2015/04/22_1	7	0.60	17.50	58.55
2015/04/22_1	8	0.02	17.74	58.76
2015/04/22_1	9	0.64	17.74	58.76
2015/04/22_1	10	0.43	17.33	58.87
2015/04/22_1	11	0.47	18.02	58.96
2015/04/22_1	12	0.83	17.22	58.24

race_key	actual champ dr	pred champ dr	Hit
2015/04/22_1	1	5	FALSE

Fig 4.1.1: Purpose and workflow of Modeling

The first assumption (asm1) is that each entry is independent with each other, and the result, either speed or winning probability only relies on the regressors. Then, we conducted some initial trials, and found that binary classification would be more stable across models, and have good interpretability. Hence, we'll **focus on binary classification afterwards for asm1**, with cost as:

$$J(h_\theta, D) = -\sum_{(x_i,y_i)\in D} y_i \times \log\big(h_\theta(x_i)\big) + (1-y_i)\times\log\left(1-h_\theta(x_i)\right)$$

Where x_i and y_i is the feature and label of each candidate from the dataset D, $h_\theta(x)$ is the output score of the model, either ML or DL, given parameters θ and x_i. Such loss may be different for some tree-based models, but the task stays the same.

Besides, inspired by pairwise Learning to Rank (Tie-Yan Liu et al., 2009), we can further capture the pairwise order between candidates, with the underlying **assumptions (asm2)** that the score of each candidate is dependant on each other, thus a joint training for pairs of entries is made possible. Based on asm2, we can draw the pairwise ranking cost function as:

$$J(h_\theta, D) = -\sum \log\sigma(h_\theta(x_i) - h_\theta(x_j))$$

The model focus on enlarging the gap between the score between $h_\theta(x_i)$ and $h_\theta(x_j)$, as long as x_i has a higher rank than x_j.

For machine learning (ML) based models, we can safely implement with asm1, but hard to do with asm2, since sklearn do not provide us with such modification, while for deep learning based-models, we would compare both.

Note that we would tune the hyper parameters of interest for all models in each experiment, so that the best performance is captured. To sum up, our plan overall offset would be:

Model	Assumption		Numeric Features	Categorical Features			Hyperparameter
Type	ASM1	ASM2	Use Best	One-Hot	Descriptive	Embedding	Tuning
ML Models	✓		✓	✓	✓		✓
DL Models	✓	✓	✓		✓	✓	✓

Fig 4.1.2: Modeling Offsets

4.2 ML Models

ML models have highly effect on classification which the data is the large panel data. Our group have chosen five ML models to do the training, validation, and testing under asm1.

1. Logistic Regression

Logistic regression analysis is a generalized linear regression analysis model, which belongs to supervised learning in machine learning. The derivation process and calculation method are similar to the process of regression, but in reality, they are mainly used to solve binary classification problems (and can also solve multi classification problems). For our problem, it is a kind of binary classification problem: champion or not. So we choose this model as our baseline model.

- Main parameters: max_iter, C

2. Decision Tree Classification

The decision tree is a decision analysis method to calculate the probability that the expected value of net present value is greater than or equal to zero, evaluate the project risk and judge its feasibility based on the known probability of occurrence of various situations, and it is a graphical method to intuitively use probability analysis.

- Main parameters: criterion, splitter, max_depth, min_samples_leaf, min_samples_split

3. Random Forest Classifier

Random forest is an algorithm that integrates multiple trees through the idea of integrated learning. Its basic unit is the decision tree, and its essence belongs to a major branch of machine learning - ensemble learning.

- Main parameters: criterion, max_depth, min_samples_leaf, min_samples_split

4. AdaBoosting Classifier

Each classifier integrated by AdaBoosting is the same, and each classifier has a high accuracy for a specific range of data. Some cannot be correctly classified and placed in the next round, and the weight of misclassified data is increased. This prompts the next round of classifiers to focus on identifying the wrong data in this round, thereby improving the classifier's ability; After several rounds, a set of classifiers will be obtained, and inputting data into this set of classifiers will result in a comprehensive and accurate classification result.

- Main parameters: n_estimators, learning_rate, algorithm

5. XGBoost Classification

XGB (extreme gradient boosting) is an industrial implementation of GBDT. It also reduces the loss function by adding new trees and fitting pseudo residuals. The fitting process is the second order Taylor expansion of the loss function, which is different from GBDT.

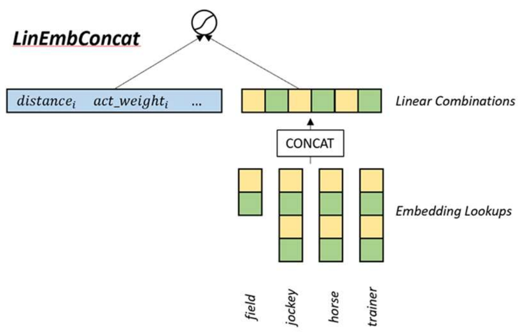
- Main parameters: colsample_bytree, reg_alpha, reg_lambda, max_depth, min_child_weight, subsample, n_estimators, learning_rate

We have designed a class which is called HKJCmodel, this class is designed to train the model, save the model and test the model which is user-friendly.

4.3 DL Models

PyTorch provides easy implementations on deep learning models, with structure free of adjustments. As planned in 4.1, we would consider categorical features' embeddings for deep models. However, the question is how to incorporate the embeddings into deep models, e.g. concatenation, dot product, element-wise product, etc?

To tackle this problem, we design 3 elementary network architectures to verify each's efficacy, Fig 4.3.1-4.3.3 illustrates their designs:



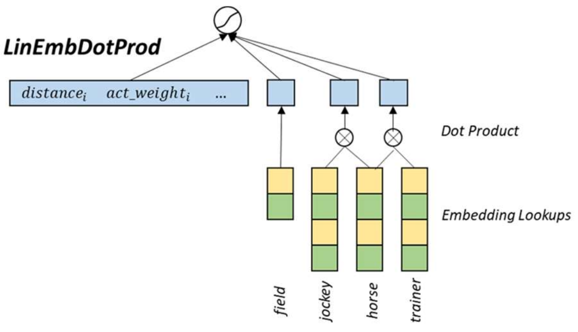
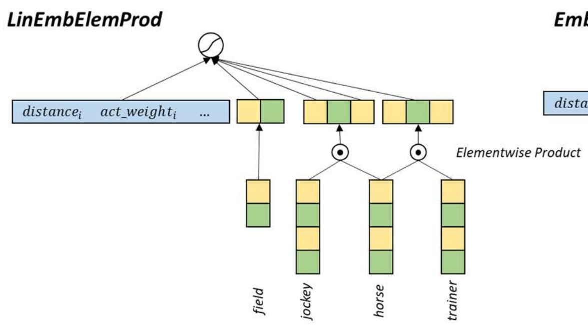


Fig 4.3.1 Concatenation of all embeddings; **Fig 4.3.2** Dot product of jockey, horse, and horse trainer



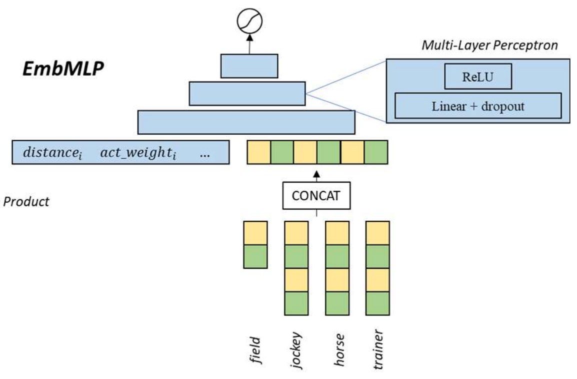


Fig 4.3.3 Element-wise product; **Fig 4.3.4** Concatenation of features + MLP

Suppose we have k dimension embedding for each entry, for each model, we have:

- **LinEmbCon**: directly concate all embeddings and pass them along with the feature, having <u># numeric features + k*# categorical features</u> dimensions.
- **LinEmbProd**: In order to reduce the dimension, also capture the interaction of each entries,

we make dot product between horse with jockey, and horse with trainer, bringing # numeric features + 2 dimensions into downstream layers.

- **LinEmbProd**: In order to capture the interaction of each entries while preserving information inside embeddings, we make element-wise product between horse with jockey, and horse with trainer, bringing # numeric features + 2*k dimensions into downstream layers.
- Parameters of interest would be learning_rate, weight_decay.

Based on these ways to process embeddings, we find that concatenation would brings about the best performance, since it greatly preserves information (please check 5.3). Based on findings, we extend the basic network structure of LinEmbCon with MLP to further capture the non-linearity pattern within data, as shown in ***Fig 4.3.4***, we add MLP over the features with concatenated embeddings:

- **EmbMLP**: Based on the concatenation of embeddings, and features, we further pass the data into MLP layers and make inference. For each linear layer inside MLP, we add dropout to turn off 10% of neurons, in order to avoid overfitting, and pass a ReLU to learn high-order patterns. Each layer afterwards may reduce the original size by 1/2, thus compress the information.

Besides learning_rate and weight_decay, we are also interested in number of layers for EmbMLP.

Note that parameters for all networks are initialized with kaiming initialization. Embedding size, number of epochs and batch size has been already carefully studied in notebooks, thus omitting them afterwards.

4.4 Pairwise Training Manner

The horse racing problem somehow relates to ranking, either score or probability, by retrieving the best one as the candidate prediction for champion. However, **previous works in the field have not pay attention to the ranking order within each game**, by just treating every candidate independently. Following the tradition, we have made ML and DL models based on it (asm1), but we soon noticed its limitation from experiment.

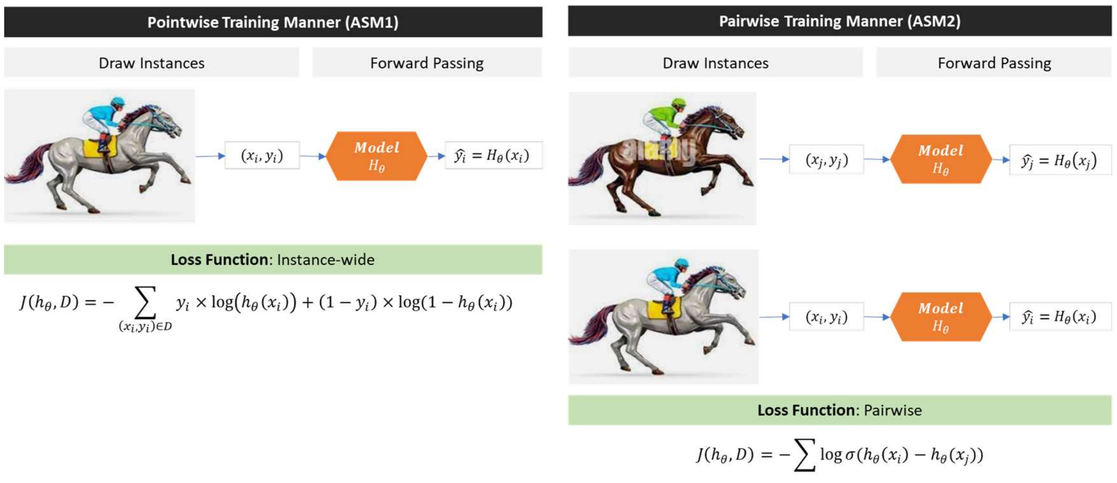

Pointwise Training Manner (ASM1)
Draw Instances
Forward Passing
(x_i, y_i)
Model H_θ
$\hat{y}_i = H_\theta(x_i)$
Loss Function: Instance-wide
$J(h_\theta, D) = -\sum_{(x_i,y_i)\in D} y_i \times \log(h_\theta(x_i)) + (1-y_i) \times \log(1-h_\theta(x_i))$
Pairwise Training Manner (ASM2)
Draw Instances
Forward Passing
(x_j, y_j)
Model H_θ
$\hat{y}_j = H_\theta(x_j)$
(x_i, y_i)
Model H_θ
$\hat{y}_i = H_\theta(x_i)$
Loss Function: Pairwise
$J(h_\theta, D) = -\sum \log \sigma(h_\theta(x_i) - h_\theta(x_j))$

Fig 4.4.1: Difference in training between pointwise (ASM1) and pairwise (ASM2)

In order to capture the ranking order, models should pay attention to multiple instances at a time. Pairwise training is trending in tasks focus on ranking, especially in information retrieval, such as recommendation (BPR, Rendel et al., 2009), NLP (QA-LSTM, Tan M et al., 2015), which usually output ranking score for each instance, and maximize the gaps between each ordered pair while achieve decent robustness [7]:.Fig 4.4.1 illustrates the differences of pairwise training manner in horse racing champion prediction and traditional pointwise manner.

Such training manner is more effective and robust compared with independent ones, since no extra modifications are made regarding the model structures while the ordering is captured.

However, the problem is, if we sample all pairs within each game, we would have too much data. Say we have N candidates in each M games, we would end up with xx pairs. Fig 4.4.2 can help us understand such dilemma. Hence, in practice, we sample with some proportion γ, e.g. 50%, to draw pairs sparsely, which can greatly reduce the size of the dataset.

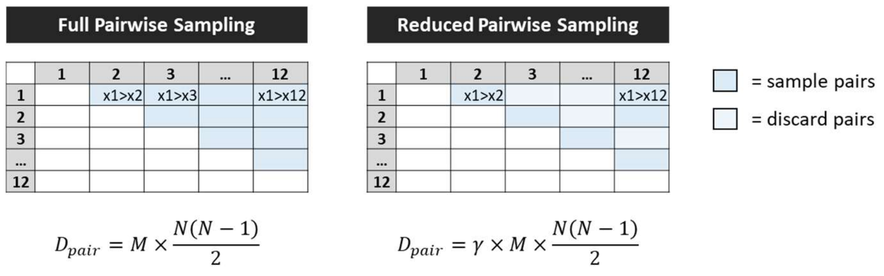

Full Pairwise Sampling
Reduced Pairwise Sampling
x1>x2
x1>x3
x1>x12
= sample pairs
= discard pairs
$D_{pair} = M \times \frac{N(N-1)}{2}$
$D_{pair} = \gamma \times M \times \frac{N(N-1)}{2}$

Fig 4.4.2 Issues, remedies of Sampling

4.5 Grid Search

In order to ensure a higher accuracy of the prediction effect, we use many models to train the data. The larger the number of models, the more tuning parameters we need. program, so we used an exhaustive method-grid search to find the most effective model parameters with good results. We independently discovered the grid search algorithm, given the entire range of parameters, tried all sorting possibilities, then selected the best parameter combination and compared it with other models, and finally selected the model with the best performance.

Notice: *All the parameters are adjusted on the AutoDL farm and HKU GPU farm and the codes should run on GPU.*

5. Experiments

5.1 Pre-processing

Due to the fact that the data is obtained from the official website of the Hong Kong Jockey Club, the level of data is not ideal. Therefore, we must **expand the race into record level for training**. For example, we will label the participants in each round of the race, and each participant is a separate piece of data. Therefore, our data can be expanded by about 10 times.

Secondly, we segment the dataset in an 8:1:1 ratio to generate training, validation, and testing sets, respectively. We will further train our model based on training set, and see models' performance given different hyperparameters on validation set, and pick the one with the highest performance to implement on the testing set.

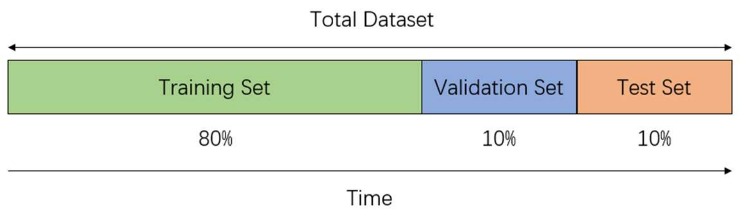


Fig 5.1.1 Dataset Split Logic

Besides, since some models are sensitive to the scale of the data, for example, logistic regression with L2 norm, and other deep learning models with AdamW in L2 norms, we need to **standardize all numerical features**, so that models can treat the features equivalently:

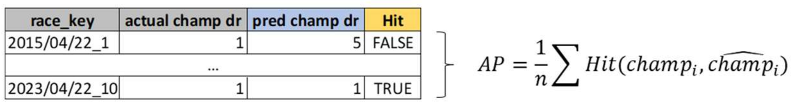
$$x^* = \frac{x - mean(x)}{std(x)}$$

All these processing steps are available in load_data module, and train modules in our repository.

5.2 Metric, and Performance Analysis

In order to evaluate the performance of all models, we **propose Average Precision (AP) as our metric** and **2 baselines for fair comparisons**.

By nature, each champion must be the fastest one in the game among his competitors, usually 11 per race. Hence, metrics like accuracy is not ideal, since, by nature, it would be high due to the imbalance of label. On the other hand, we need to make sure that for each game, each strategy only provides one prediction as champion. Hence, we propose **Average Prediction (AP)** as our metric. By simply computing the hit rate of champion prediction, we can measure to what extent the prediction would win.



race_key	actual champ dr	pred champ dr	Hit
2015/04/22_1	1	5	FALSE
	...		
2023/04/22_10	1	1	TRUE

$$AP = \frac{1}{n}\sum Hit(champ_i, \widehat{champ_i})$$

Fig 5.2.1 Intuition of Average Precision (AP)

Moreover, we need baselines to measure how good the AP is:

- **Baseline 1- Random Guess**: By randomly picking one candidate as the champion, repeating draw 1000 times, and computing their sample average, we have AP at 8.6% on testing set, indicating we would have 8.6 hits on champion out of 100 races out of luck.
- **Baseline 2- Following the lowest winning odds**: Another consideration would be follow the trend. The winning odds somehow reflect people's belif on the result, the lower the winning odds, the more probable people think the candidate would win. Taking the one with the lowest winning odds as guess, we reach AP at 28.1%, indicating we would hit the champion 28.1 times out of 100 races.

One constraint is the winning odds scrapped online is the closing odds, which is unaccessible before making bids. Also, we intend to make prediction better than the winning odds. Hence, we would **remove winning odds while making predictions**.

In order to draw models' champion guesses, we pick the candidate with the highest winning probability, or scores in each game. By repeating the step across each race in validation, and testing set, we have the according guesses.

By exhausting all possible parameters with grid search, picking the ones with the highest validation AP, and evaluating their the performance on testing set, we have the performance table as follows. Since validation set is mainly for hyperparameter tuning, we would mainly lay our focus on the performance across models in testing set. Note that all models mentioned are tunned to the best:

Baselines	**Validation Performance Average Precision (AP, %)**			**Testing Performance Average Precision (AP, %)**		
Baseline 1: Random Guess	8.6%	-	-	8.6%	-	-
Baseline 2: Follow the Lowest Odds	32.4%	-	-	28.1%	-	-
Model (With out Feature win_odds)	**AP(%)**	**Imprv v.s. b1 (Percent Point)**	***Imprv v.s. b2 (Percent Point)***	**AP**	**Imprv v.s. b1 (Percent Point)**	***Imprv v.s. b2 (Percent Point)***
Logistic Regression	25.6%	17.1 pp	-6.8 pp	22.7%	14.1 pp	-5.4 pp
Decision Tree	25.8%	17.2 pp	-6.6 pp	20.3%	11.7 pp	-7.8 pp
Random Forest	25.4%	16.8 pp	-7.0 pp	22.9%	14.4 pp	-5.2 pp
AdaBoost	30.1%	21.5 pp	-2.3 pp	22.3%	13.7 pp	-5.8 pp
XGBoost	**31.8%**	23.2 pp	-0.6 pp	**23.3%**	14.8 pp	-4.8 pp
LinEmbConcat	23.5%	14.9 pp	-8.9 pp	22.3%	13.7 pp	-5.8 pp
LinEmbDotProd	17.7%	9.1 pp	-14.7 pp	17.7%	9.1 pp	-10.4 pp
LinEmbElemProd	22.2%	13.6 pp	-10.2 pp	19.8%	11.2 pp	-8.3 pp
EmbMLP-Layer2	24.2%	15.6 pp	-8.2 pp	22.3%	13.7 pp	-5.8 pp
EmbMLP-Layer5	**26.1%**	17.5 pp	-6.3 pp	**23.3%**	14.7 pp	-4.8 pp
LinEmbConcat - pairwise	27.5%	18.9 pp	-4.9 pp	23.3%	14.7 pp	-4.8 pp
LinEmbDotProd - pairwise	19.5%	10.9 pp	-12.9 pp	21.0%	12.4 pp	-7.1 pp
LinEmbElemProd - pairwise	**29.0%**	20.4 pp	-3.4 pp	22.9%	14.3 pp	-5.2 pp
EmbMLP-Layer5 - pairwise	28.6%	20.0 pp	-3.8 pp	**23.4%**	14.8 pp	-4.7 pp

Fig 5.2.3 Performance of all models on validation and testing set

From the **machine learning** models we can find that on the validation set XGBoost model has the best performance (AP: 31.8%) among five models which is same on the test set (AP: 23.3%). Because ML models are trained without odds so that according to the baseline1 our model performance has increased much more than the baseline model. Meanwhile, other models also have good performance like AdaBoost model (AP: 30.1% on validation set; AP: 22.3% on test set).

On the other hand, for the **deep learning models trained under pointwise manner (asm1)**, EmbMLP with 5 MLP layers perform the best in validation, and testing AP (26.1%, 23.3%), which equally performed the XGBoost in testing set. However, the rest models' performances are not that desirable, with few approximating tree-based ensemble models.

Concatenation would be the best to aggregating embeddings, since it restores the most information, and can thus achieve better AP while training, which one can easily found from the table among the performance of LinEmbConcat against DotProd's and ElemProd's. Hence further updates based on concatenation was implemented to EmbMLP.

Deep learning models trained under pairwise assumptions (asm2) achieved significant improvement over pointwise ones. The highest test AP is achieved by EmbMLP with 5 layers at 23.4%. What is more, all architectures achieved higher AP performances on both validation and testing set, indicating **asm2 is much more suitable for horse racing champion prediction**.

5.3 Key Questions & Findings

Based on all experiments and performances, we can now answer some interesting questions.

KQ 1. Have we beat baselines? Which model performs the best? Any reasons behind?

After hyperparameter tuning, **our models all outperformed the random baseline**, indicating that the features distilled, assumptions made, and models built can learn the pattern to some extent, with AP at most 23.3% by XGBoost on testing set. However, unfortunately, **none of the models perform better than baseline 2**. Current assumptions are not effective enough to beat winning odds. Moreover, deep learning-based model EmbMLP does not stably outperform all tree-based models, with 5.7pp and 0.4pp gap of AP compared with XGBoost.

Wondering the reasons behind, research in NIPS (Léo Grinsztajn et al., 2022) proposed some possible reasons behind, after comparing performances of ResNet, Transformers, against XGB, AdaBoost, and RF, from 40 benchmark tabular datasets. They argue that **neural networks like MLP are easily affected by the uninformative features**, since they are not rotationally invariant (Andrew Ng, 2004) to data while training, **while tree-based models by nature treat features separately**, hence treating each features' information more seriously, and avoiding the disturbance of uninformative ones.

KQ 2. How to treat categorical features? Are feature engineering helpful?

In 3.6, we proposed 3 treatments for handling the categorical features: one-hot encoding, information description, and embedding. In deep learning models, we are capable to implement and test the effectiveness of the descriptive encoding, and embedding.

To evaluate this, we pass different subsets of features into all deep learning models, and check their ability in improving AP in validation and testing set:

- **Subset1 - ID**: Only categorical features would be passed, and models would only consider embeddings and learn weights to infer the probabilities of win.
- **Subset2 – Full+ID**: In this subset, categorical features, and the full set of features (100+) would be passed to the model.
- **Subset3 – Best+ID**: Best 22 features, instead of the full set will be passed to the model along with categorical features. Model would learn from the most fine-grained set of features.

After grid searching with the best hyperparameter of each model, the best performance under different input for each model would be:

DL Model	Features	Validation Performance		Test Performance	
		AP(%)	Imprv v.s. ID (Percent Point)	AP	Imprv v.s. ID (Percent Point)
LinEmbConcat	ID	21.8%		19.6%	
	Full+ID	21.8%	0%	20.4%	4%
	Best+ID	23.5%	8%	22.3%	14%
LinEmbDotProd	ID	13.2%		13.2%	
	Full+ID	11.0%	-17%	16.4%	24%
	Best+ID	17.7%	34%	17.0%	29%
LinEmbElemProd	ID	16.3%		17.3%	
	Full+ID	17.7%	9%	17.7%	2%
	Best+ID	22.2%	36%	19.8%	14%
EmbMLP-Layer2	ID	23.5%		17.7%	
	Full+ID	16.2%	-31%	21.0%	19%
	Best+ID	24.2%	3%	22.3%	26%
EmbMLP-Layer5	ID	26.1%		21.9%	
	Full+ID	21.9%	-16%	20.7%	-6%
	Best+ID	27.1%	4%	24.8%	13%

Fig 5.3.1 APs for deep-learning-based models given each subset of features

From Fig5.3.1, we can see that:

1) **Models' performance with only categorical embeddings is good enough**, especially for deep MLPs. Further adding numerical features do help improve the performance, but the APs surrounds the one with only categorical inputs.

2) **Feature engineering helps improving AP**. Regarding the testing APs, those with more features generated from engineering, would boost performance up to nearly 30% compare with only categorical inputs, indicating our features do have much relationship with the target.

3) **Feature selection helps reduce uninformative features, thus benefits the overall performance**. Those with Best+ID typically has large increase against the pure categorical, or with the full features. Besides, the testing performance is higher than other kinds of inputs, which means deep learning models are less distracted by uninformative features, and achieve more robust performance overall.

4) **Adding layers to MLP do helps, but the marginal benefits decreases** as well.

KQ 3. Which features are useful for champion prediction?

Among all models we tuned, the logistic model is good enough for champion prediction, since it has decent AP on both validation and testing set. Moreover, the interpretability of Logistic model is unimpeachable.

From Fig5.3.2, one can observe the parameters of Logistic Regression. After removing the standard deviation of each regressors, we may find the most important features, and their underlying information, for example:

- **Jockey's winning rate within the past year**: for every 1 percent point increase in the winning rate, the probability of winning over not win would increase 6.6%.
- **Times horses ranked top4 within last 5 races**: for every 1 time increase in the times horses' ranked top4 within last 5 races, the probability of winning over not win would

increase 56.4%.

Similarly, other interpretations could be made under similar manner.

Feature	Betas (Raw)				Betas (Removed Std)	
	value	mean	variance	Effect on Odds	value	Effect on Odds
jockey_champ_rate_y	0.494	0.085	0.006	63.9%	0.064	6.6%
horse_top4_last5	0.135	0.309	0.091	14.5%	0.448	56.4%
horse_top4_rate_y	0.087	0.264	0.111	9.1%	0.003	0.3%
jockey_top4_rate_m	0.049	0.318	0.03	5.0%	0.003	0.3%
horse_place_m	0.042	0.135	0.124	4.3%	0.119	12.7%
jockey_top4_rate_y	0.034	0.316	0.024	3.5%	0.002	0.2%
trainer_champ_rate_h	0.034	0.086	0.001	3.5%	0.011	1.1%
horse_champ_rate_h	0.029	0.08	0.019	2.9%	0.002	0.2%
jockey_champ_y	0.021	12.677	347.432	2.1%	0.001	0.1%
trainer_top4_rate_y	0.013	0.323	0.011	1.3%	0.001	0.1%
jockey_top4_m	0.01	10.323	68.614	1.0%	0.001	0.1%
jockey_elo	0.009	22.486	9493.47	0.9%	0.000	0.0%
trainer_champ_rate_y	0.009	0.082	0.002	0.9%	0.002	0.2%
trainer_place_rate_h	0.007	0.252	0.004	0.7%	0.001	0.1%
trainer_top4_rate_m	0.003	0.324	0.016	0.3%	0.000	0.0%
race_money	-0.004	1452157.754	5.26187E+12	-0.4%	0.000	0.0%
horse_champs_h	-0.004	0.874	1.585	-0.4%	-0.003	-0.3%
horse_top4_h	-0.01	3.45	15.272	-1.0%	-0.003	-0.3%
horse_life_time_d	-0.013	433.459	143131.03	-1.3%	0.000	0.0%
jockey_champ_last5	-0.017	0.083	0.018	-1.7%	-0.127	-11.9%
jockey_champ_rate_h	-0.025	0.086	0.004	-2.5%	-0.004	-0.4%
horse_bestperform_h	-0.086	2.798	9.066	-8.2%	-0.029	-2.8%

Fig 5.3.2 Parameters of Logistic Regression model

KQ 4. Are Asm2 more suitable for this condition?

DL Model	Features	Pointwise (ASM1)		Pairwise (ASM2)		Imprv (%)	
		Validation AP (%)	Testing AP (%)	Validation AP (%)	Testing AP (%)	Validation	Test
LinEmbConcat	ID	21.8%	19.6%	28.6%	22.9%	31.2%	16.8%
	Full+ID	21.8%	20.4%	27.1%	22.7%	24.3%	11.3%
	Best+ID	23.5%	22.3%	27.5%	23.3%	17.0%	4.5%
LinEmbDotProd	ID	13.2%	13.2%	15.7%	16.9%	18.9%	28.0%
	Full+ID	11.0%	16.4%	17.4%	15.8%	58.2%	-3.7%
	Best+ID	17.7%	17.0%	19.5%	21.0%	10.2%	23.5%
LinEmbElemProd	ID	16.3%	17.3%	29.0%	22.7%	77.9%	31.2%
	Full+ID	17.7%	17.7%	28.4%	23.1%	60.5%	30.5%
	Best+ID	22.2%	19.8%	27.8%	22.9%	25.2%	15.7%
EmbMLP-Layer5	ID	26.1%	21.9%	27.8%	23.1%	6.5%	5.5%
	Full+ID	21.9%	20.7%	27.8%	22.3%	26.9%	7.8%
	Best+ID	27.1%	23.3%	28.6%	23.4%	5.6%	0.4%
Average						**30.2%**	**14.3%**

Fig 5.3.3 Improvement of pairwise training over pointwise

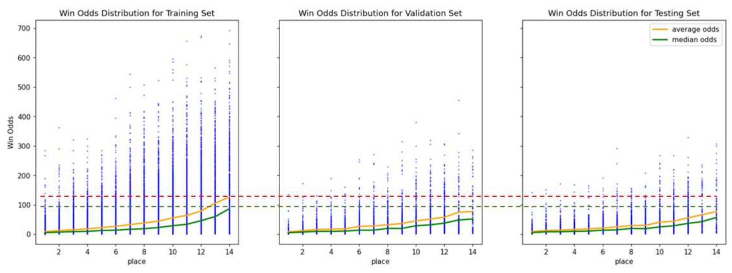
The answer is obvious. From Fig 5.3.3, nearly all models trained under pairwise manner have lot improvements regarding AP compared with ASM1, on average 30% improvement on validation, and 14% on testing set. Such improvements suggest that **we have found a better training manner in this task, compared with works beforehand**.

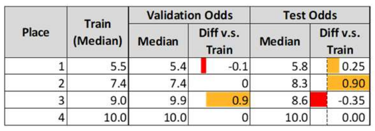
KQ 5. Why Performance degrades in testing set? How to improve?

In our result accuracy display, we can find that there is a huge gap regarding AP for both baseline2, and all models built. All APs degrade from validation set to testing set. In order to seek the reasons behind, we again plot the win odds against each place, as shown in Fig5.3.4:

We observed that, the distribution of win odds for each place on the training set, test set, and validation set is little different:

1. Win odds on the training set is comparatively more disperse against validation and testing set, while the later two sets are more alike.
2. Top-1 win odds on testing set is much higher than we expected, indicating some abnormal changes in the distribution of the data recently.





Place	Train (Median)	Validation Odds		Test Odds	
		Median	Diff v.s. Train	Median	Diff v.s. Train
1	5.5	5.4	-0.1	5.8	0.25
2	7.4	7.4	0	8.3	0.90
3	9.0	9.9	0.9	8.6	-0.35
4	10.0	10.0	0	10.0	0.00

Fig 5.3.4 Distribution of Winning Odds across training, validation, and testing set

With these findings, we may answer why baseline2 is much lower. Also, with regard to the changes happened in short-term, our models may be failed to capture them, thus resulting in AP degradation.

6. Conclusion, Limitations and Future works

To conclude, we have performed a closed data mining procedure, including data collection, pre-processing, feature engineering and selection, modelling, evaluation, and analysis. The whole process of HKJC racing prediction has never been done so complete before, and we are lucky to be the first to publish the all-round pipeline to the public network.

In feature engineering, we built 100+ features including statistical and algorithmic, and rigorously select 22 features out of them with good predictabilities. Further experiments show how selection is worthwhile 1) across ML and DL models in removing uninformative features; 2) for better interpretability for analysis.

Previous works lay huge efforts on pointwise estimation of winning (Chuen W.C. et al, 2017), which ignores the order among participants. Inspired by the trending technique of pairwise ranking, we pioneeredly introduce, and applied it into horse racing champion prediction. The significantly improved performance across all models indicating this was a better training manner over pointwise.

However, there also exit limitations in our work, such as the bias of ELO features. ELO features rely on historical data of the performance of horses, jockeys or trainers in the past races, and if a horse, jockey or trainer participates more frequently in races, it may have more opportunities to accumulate wins and improve its ELO scores.

For future works, we may focus on the following directions:

1) **Incorporating new racing data as mini-batches into model training**. As talked in KQ5, model performance degrades, since the underlying distribution have changed recently, and our training set mainly focus on long time ago. Such training manner may prevent model to learn up to date. Hence, it is an necessary step to go.
2) **Further interactions among features**. We so far found that interaction of embedding would not improve the performance on AP, but it is possible to combine the linearity of our architecture, along with the interactions with feature. In works of DeepFM (Guo H. et al, 2017), which incorporated the idea of Wide&Deep, a work of art combining linear and deep part with interaction, has achieved huge improvements over tradiciotnal linear models and fatorization models, and is worth of further trials.
3) Besides pairwise assumption, LTR also mentioned **listwise manner**, which is approximately a seq2seq framework, which we can also do some experiments on,

References

1) Liu, T. Y. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, *3*(3), 225-331.

2) Ng, A. Y. (2004, July). Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning* (p. 78).

3) Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on tabular data?. *arXiv preprint arXiv:2207.08815*.

4) He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

5) Tan, M., Santos, C. D., Xiang, B., & Zhou, B. (2015). Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

6) Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2012). BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

7) Deng, L., Lian, D., Wu, C., & Chen, E. (2022). Graph Convolution Network based Recommender Systems: Learning Guarantee and Item Mixture Powered Strategy. *Advances in Neural Information Processing Systems*, *35*, 3900-3912.

8) Chung, W. C., Chang, C. Y., & Ko, C. C. (2017, August). A svm-based committee machine for prediction of hong kong horse racing. In *2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media)* (pp. 1-4). IEEE.

9) Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... & Shah, H. (2016, September). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 7-10).

10) Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247*.