Horse Racing Data Visualization and Analysis

Chen Xiangyue P.1-7,13-15,28, Poon Sze Sen P.18.5-25,27

Wang Mingxun P.8-12,16-18.5,26

The three of us have put tremendous effort into this project and we want a group grading policy to be fair if possible.

Abstract

Horse racing is like a religion in Hong Kong, whose citizens bet more than anyone else on Earth. Searching for a positive return from the Hong Kong Jockey Club has always been a goal. The report will present a panoramic view of the races, the traditional analyses of the trainers and jockeys, and the methodologies to quantify horses' performances. In the end, the study provides a prediction model for racing results and a potential sure-win strategy.

1 Introduction

As an intriguing sport, horse racing has warranted the attention of Hong Kong citizens for decades-long. The excitement of horse racing was brought not only by the sprinting steeds but also the pari-mutuel wagering system. The unfathomable outcome of horse racing has always been stimulating gambler's adrenaline rush. By collecting and analyzing previous horse racing results, we attempt to construct a data visualization app and apply machine learning skills to predict future racing outcomes. With sufficient information included, it seems that long-term betting strategies can be formed to earn profits.

2 Data Preparation

Historical horse racing data are available on HKJC's official website. We constructed a huge database by scraping the raw data and organizing them into several Excel files. It consists of the records of races, horses and trainers over a span of 13 years (2008 to 2020). The organized database allows us to do further processing, perform data visualization and build the model:

The "racing record" dataset involves all racing data from 2008 to 2020. Performance of a certain horse in a particular race is recorded in each row. The dataset contains 117800 records from the races taken place in Hong Kong. The "races" dataset contains general information of 9255 races taken place in Hong Kong from 2008 to 2020. Each row in the dataset represents a particular race. The "horses" dataset contains information about 6167 horses that ran in race from 2008 to 2020. Some selected characteristics from this dataset could help our model distinguish between different horses. The "trainers" dataset records the performance of 213 trainers who were active from 2008 to 2020. By recording the winning percentage of each trainer, the model can compare each trainer's performance. We performed data wrangling on the datasets for later visualization work:

Feature selection: Up to 40 unique features on horse racing are presented in the database. However, some of the features did not help that much to our research and some features need to be combined and transformed for plotting graphs and building the model. Therefore, only useful features are selected from each dataset and combined into new data frame for further processing.

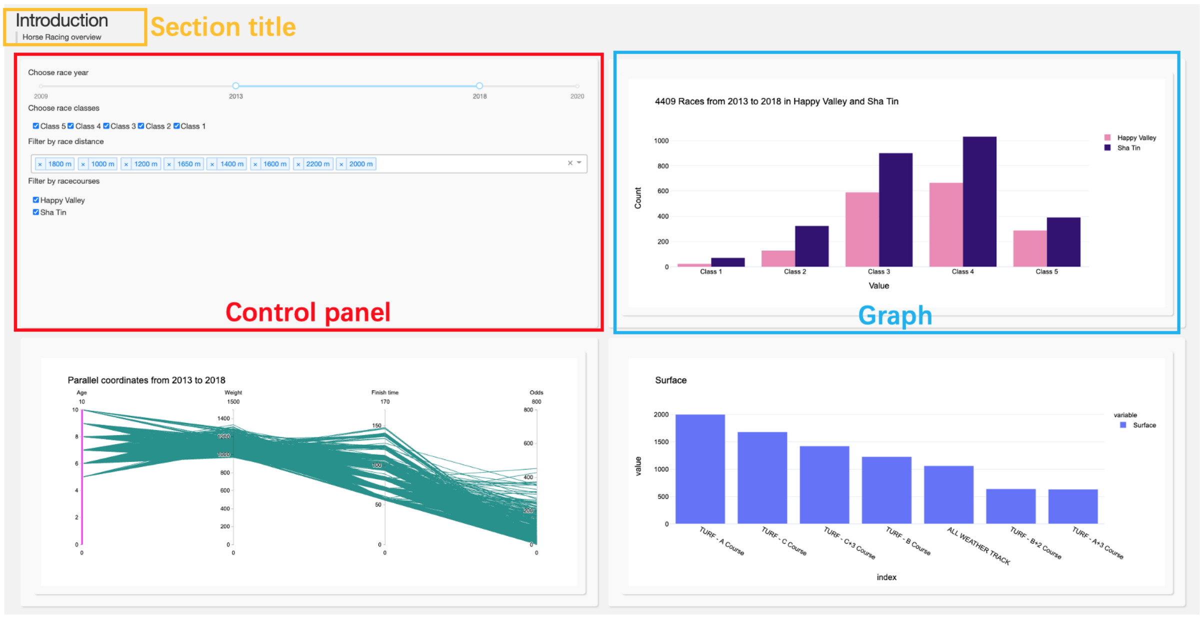
Data cleaning: Null values may present in each dataset and some are marked by "—". Those null entries are replaced by the NaN value. For simplicity, we drop the NaN values when treating with a specific feature.

Merging and transformation: Merging is the most frequently used data wrangling strategy when building our data visualization App. Datasets can be merged and transformed into the desired format for visualization according to their common columns. The races dataset is joined into racing record dataset by common column "Race_ID"; the trainer dataset is joined into racing record by "Trainer_name"and the horse dataset is merged with racing record dataset by "Horse_name".

3 Racing Overview

Every Wednesday and Sunday, races are held in either one of the venues: Happy Valley and Sha Tin. The racing distances mainly range from 1000m to 2200m. Horse racing in Hong Kong follows a neat rating scheme. Every horse that runs in the race in Hong Kong has a unique rating which is evaluated by the HKJC. A horse with a lower rating score means that it has lower quality. For instance, a horse rated 100 runs much faster than another one rated 30. Racing competitions are grouped by classes, scaling from Class 1 to Class 5. Class 1 races are competitions for horses with a rating score of 85-120. Class 5 races are competitions for those low-qualified ones rated between 0-40. There are two kinds of tracks, turf and all-weather tracks. The alphabets A, B, C indicate the distances between the inner rail and the outer rail. A course has the longest distance with 30.5m, C course has the shortest distance of 18.3m.

The introductory section in the App gives out an overview and some basic statistics on horse racing. Figure 1 demonstrates the overall structure of the introduction section of the App. Under the section title is a control panel, where users can filter out the information by selecting year range, race class, racing distances, and racecourses. Besides are the interactive graphs.



Choose race year: 2009 – [✓ 2013 – 2018] – 2020

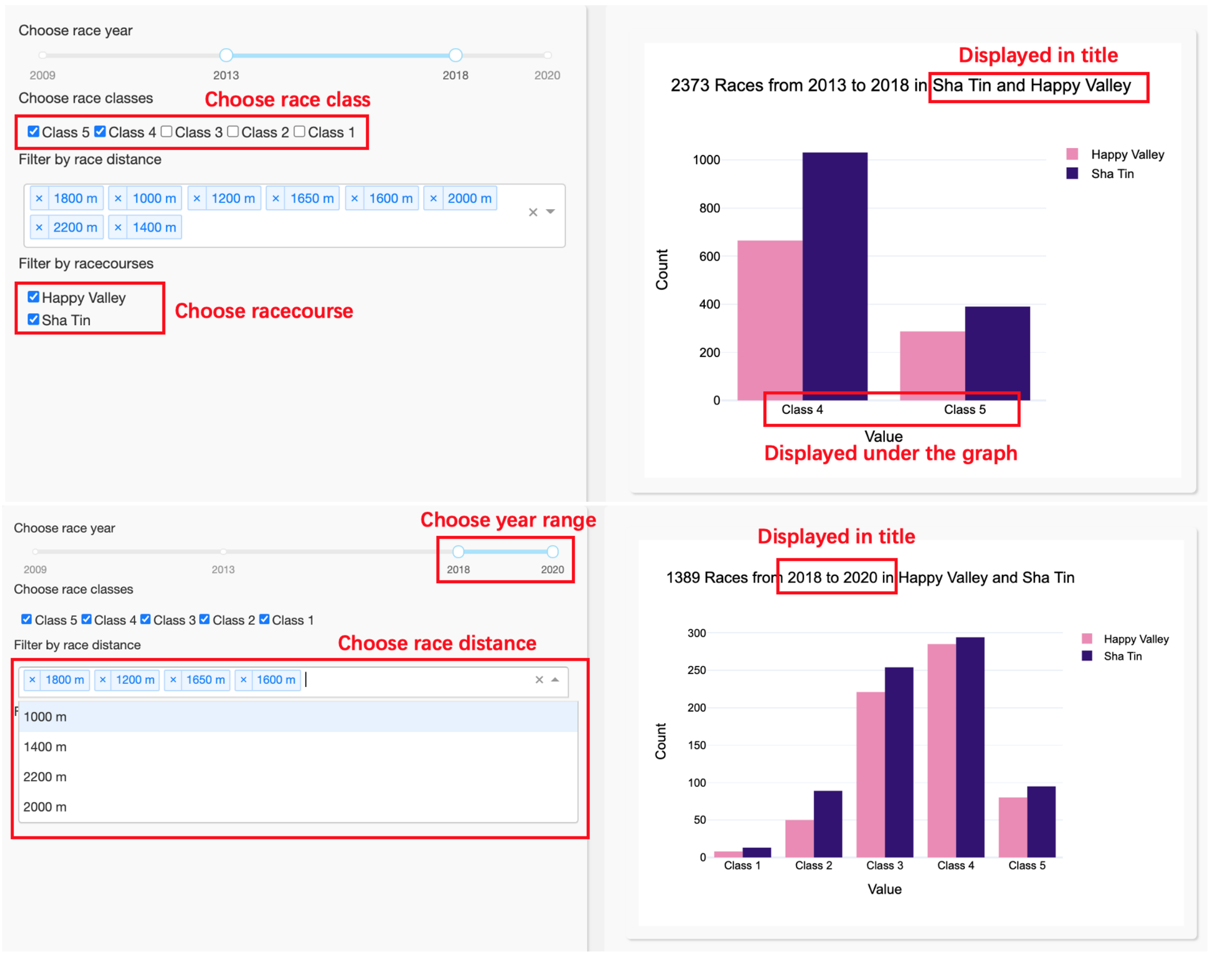
Choose race classes: [✓ Class 5] [✓ Class 4] [✓ Class 3] [✓ Class 2] [✓ Class 1]

Filter by race distance: 1800 m, 1000 m, 1200 m, 1650 m, 1400 m, 1600 m, 2200 m, 2000 m

Filter by racecourses: [✓ Happy Valley] [✓ Sha Tin]

Figure 1: Introduction section in the App

Figure 2 shows a bar chart of the number of races. By applying or dropping different filters, users can make a multi-dimensional comparison on the number of races between different race classes, race years, and racecourses. It can be concluded that the majority of racing competitions in Hong Kong lies in Class 3 and Class 4. Figure 3 illustrates information on the racing surface. Figure 4 is a parallel coordinate graph that shows the general horse racing attributes clusters. After selecting the year span, we can see the horses are mainly 5 to 10 years old. Most of them weigh between this range. The segregated clusters of finishing times represent the various finishing distances of races, ranging from 1000 to 2200m. Finally, The messiness of the lines connecting the odds column reflects the unpredictability of horse racing results. The odds can be higher than 400, representing a sure loss and as low as a single digit.



Choose race year: 2013 – 2018

Choose race classes: [✓ Class 5] [✓ Class 4] ☐ Class 3 ☐ Class 2 ☐ Class 1

Filter by race distance: 1800 m, 1000 m, 1200 m, 1650 m, 1600 m, 2000 m, 2200 m, 1400 m

Filter by racecourses: [✓ Happy Valley] [✓ Sha Tin]

2373 Races from 2013 to 2018 in Sha Tin and Happy Valley

Choose race year: 2018 – 2020

Choose race classes: [✓ Class 5] [✓ Class 4] [✓ Class 3] [✓ Class 2] [✓ Class 1]

Filter by race distance: 1800 m, 1200 m, 1650 m, 1600 m

1000 m
1400 m
2200 m
2000 m

1389 Races from 2018 to 2020 in Happy Valley and Sha Tin

Figure 2: App demo for class overview

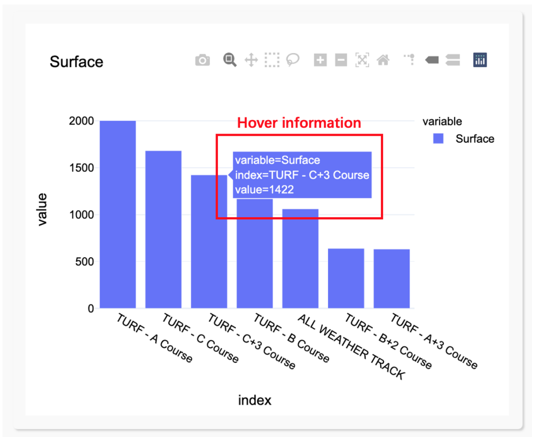


Figure 3: Surface information

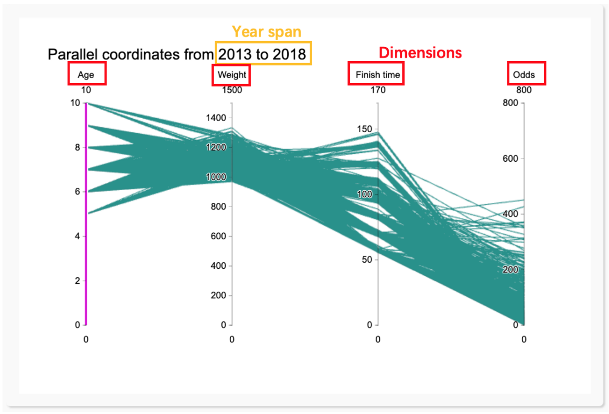


Figure 4: Clusters

4 Traditional analysis

In this section, we present some typical horse racing analyses with App demonstration. We have deployed different methodologies to quantify the performances of the horses, jockeys, and trainers. However, some of the more advanced methods that need professional data, i.e., pace handicapping, are not included in the report due to the failure of obtaining records of sectional speed.

4.1 Horse Analysis

Horse performance is an essential factor in horse racing analysis. The horse section in the App gives out some basic information on horses. Figure 6 shows a section overview. By typing in the input box, particular horses are selected and compared on different attributes. Figure 7 is an interactive world map illustrating the race horse's origins. By hovering the mouse on a particular country, a corresponding line plot about the average place of all horses in the region over the year will be shown.

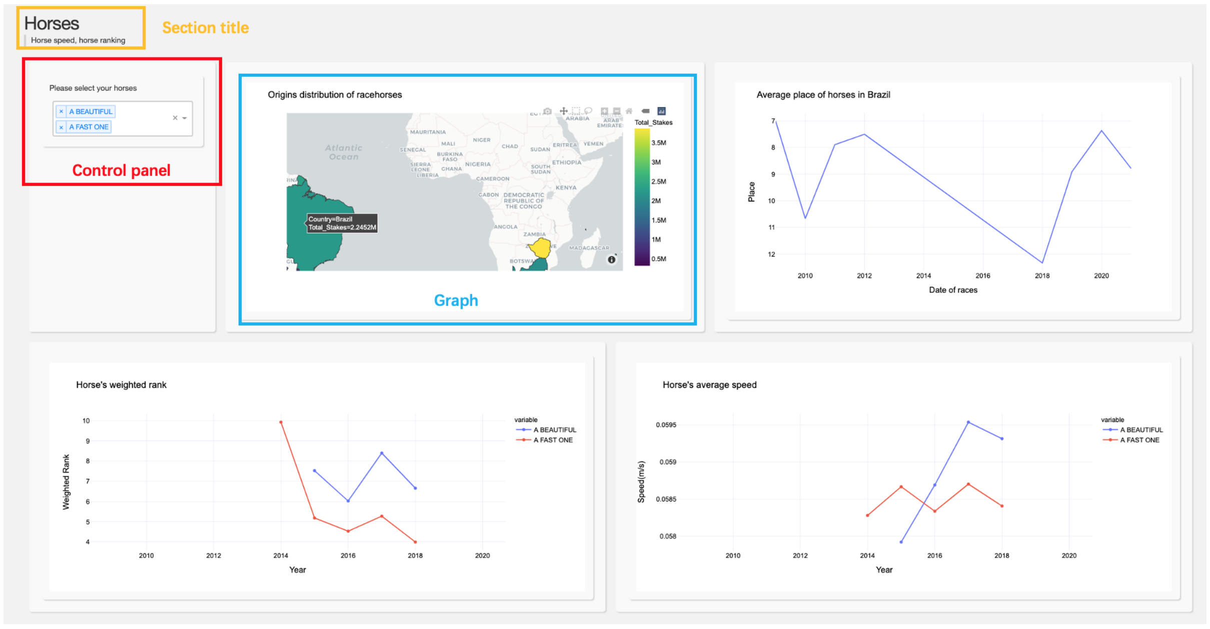


Figure 5: Overview of the Horse Panel

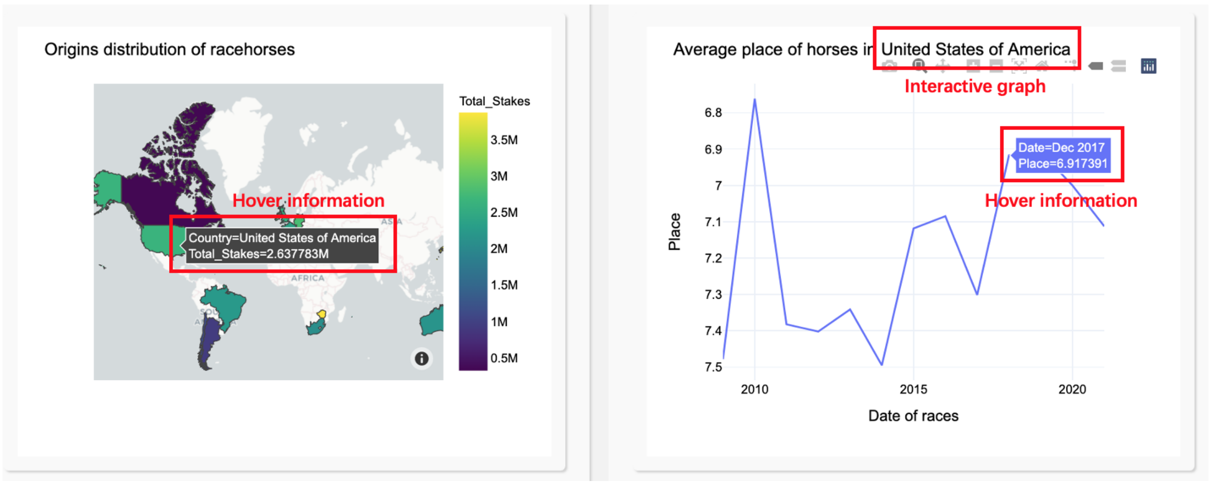
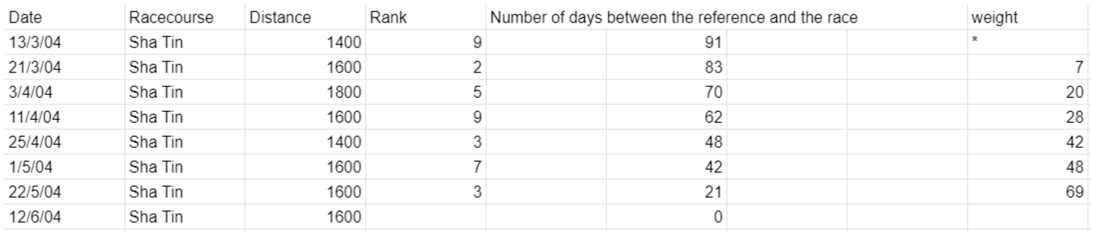


Figure 6: Racing Horse's Origins

4.1.1 Recency Weighted Rank

Below are the detailed processes of calculating the recency weighted rank of a specific horse:



Date	Racecourse	Distance	Rank	Number of days between the reference and the race				weight
13/3/04	Sha Tin	1400	9		91			*
21/3/04	Sha Tin	1600	2		83			7
3/4/04	Sha Tin	1800	5		70			20
11/4/04	Sha Tin	1600	9		62			28
25/4/04	Sha Tin	1400	3		48			42
1/5/04	Sha Tin	1600	7		42			48
22/5/04	Sha Tin	1600	3		21			69
12/6/04	Sha Tin	1600			0			

Figure 7: Performance of a random horse in 2004

Supposedly, the goal of the user is to predict the performance of this horse on 12/6/04. The first step is setting the specific date as the reference date of the analysis. It is common to include a range of 90 days to visualize the recent performance of the horse. Firstly, we should calculate the number of days between the reference date and the recent races. For instance, the earliest race is on 21/3/04, and there are 83 days in between. The same step can be applied to each of the races within 90 days. The weight is constructed using 90 minus the number of days in between. For the most recent race, the weight is $90 - 21 = 69$. The weighted rank can be calculated using the formula below.

$$\begin{aligned} Recent\ Rank &= \sum_{i=1}^{n} \frac{weight_i \times rank_i}{\sum_j weight_j} \\ &= \frac{2 \times 7 + 5 \times 20 + 9 \times 28 + 3 \times 42 + 7 \times 48 + 3 \times 69}{7 + 20 + 28 + 42 + 28 + 69} \\ &= 4.84 \end{aligned}$$

The method will emphasize recent performances and lower the weight of relatively older races. Therefore the weighted rank can serve as a reference in predicting the recent races.

4.1.2 App Demo

In the App, we use a time series plot of weighted ranks to visualize the performance of particular horses over the years. Figure 6 demonstrates the graph where users can select horses that they are particularly interested in by typing into the panel. The names of the horses are displayed in the legend of the graph.

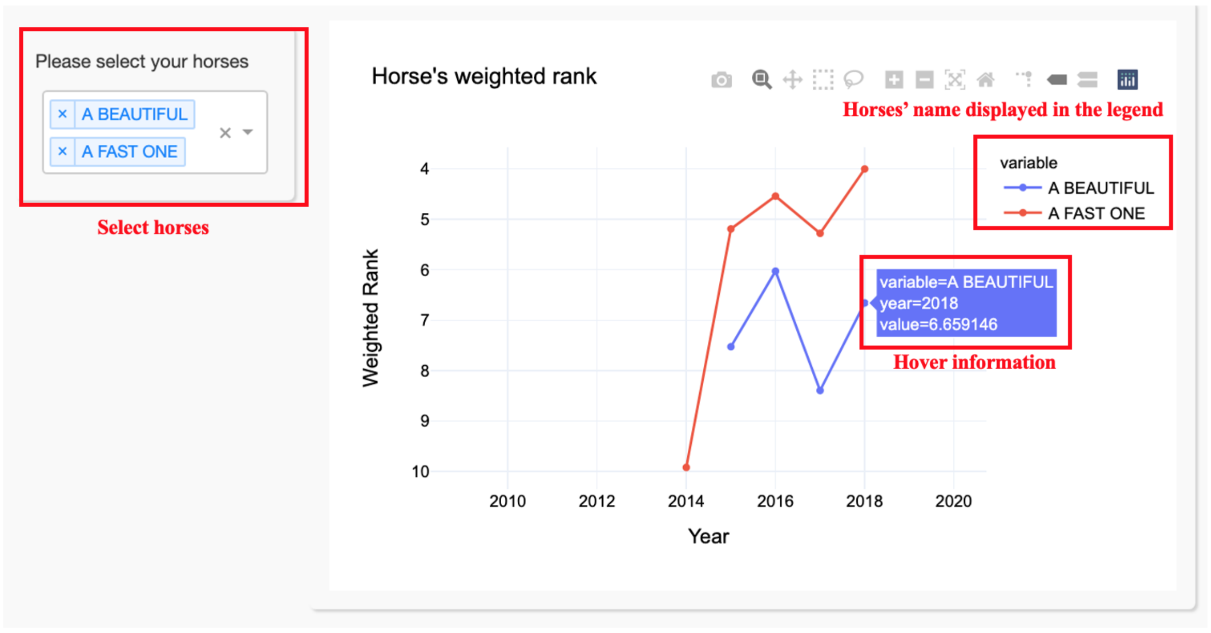


Figure 8: Horse's weighted rank

The horse's speed can be one of the most crucial factors in determining the horse's performance. By typing the horses' names in the panel, the line plot for average speed will be displayed on the right. It is useful in a way that before the bet, users can estimate the horses' performances based on the line plot. It helps users to identify the peak of the horses in their career. Higher average speed can guarantee a better result directly in a race.

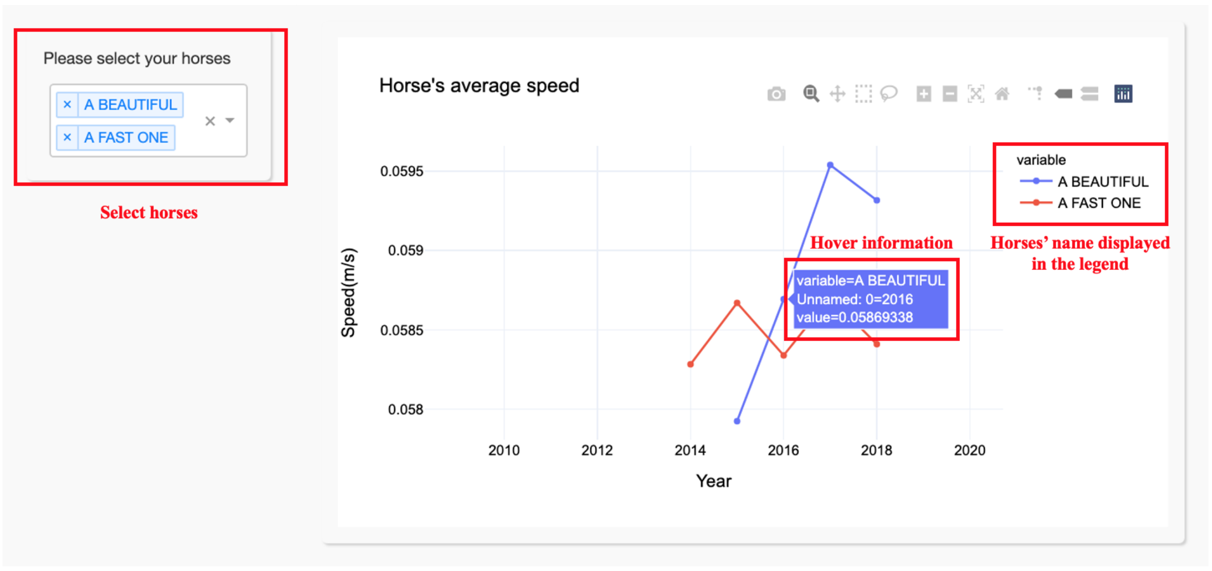


Figure 9: Horse's average speed

4.2 Performance of the Jockeys and Trainers

Jockey's performance is another crucial factor in conducting horse racing analysis. An experienced jockey will directly boost the horse's performance. Trainers are the people who are most familiar with horses. A good trainer can guide the horse to perform beyond its limit. There was no well-organized dataset recording the information of the jockeys on HKJC's website. In the end, our dataset only contains the names of the jockeys for each race. The remedial measure we have come up with uses odds as a proxy for the public's expectation of the jockeys. The jockey and trainer's winning rate is calculated and presented as a pie chart in the app to visualize the performances better.

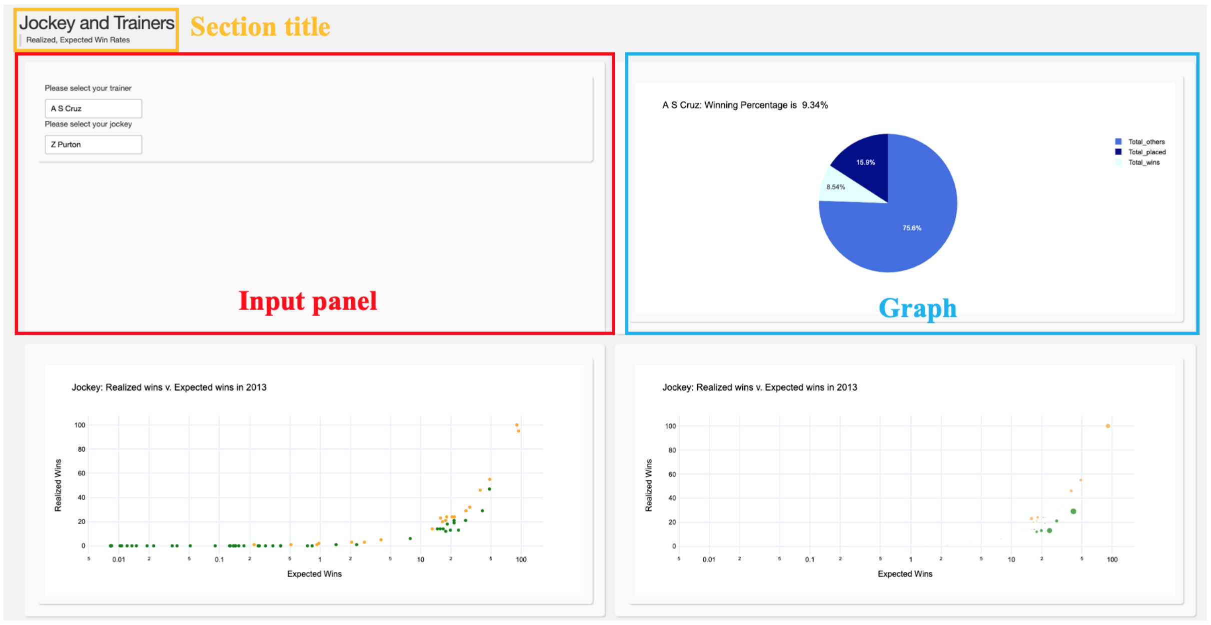
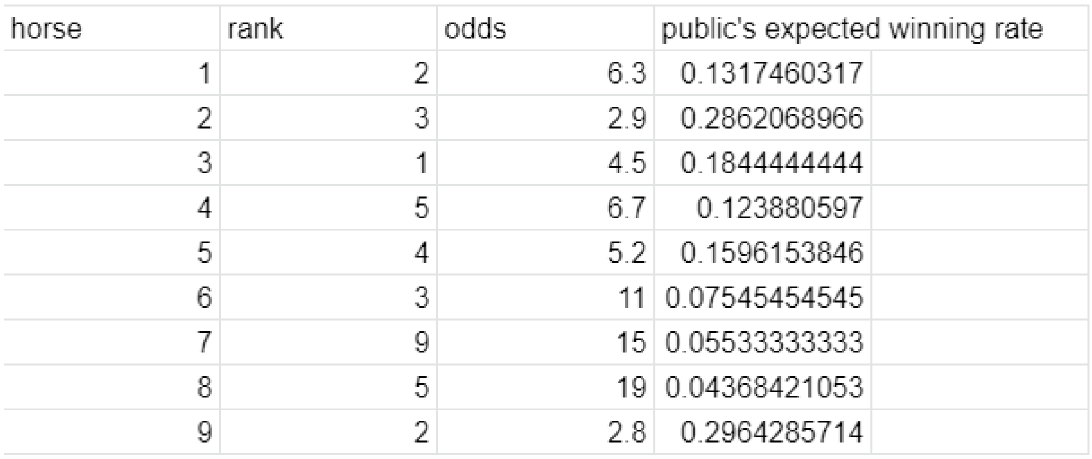

Jockey and Trainers
Realized, Expected Win Rates
Section title
Please select your trainer
A S Cruz
Please select your jockey
Z Purton
Input panel
A S Cruz: Winning Percentage is 9.34%
15.9%
8.54%
75.6%
Total_others
Total_placed
Total_wins
Graph
Jockey: Realized wins v. Expected wins in 2013
Realized Wins
Expected Wins
Jockey: Realized wins v. Expected wins in 2013
Realized Wins
Expected Wins

Figure 10: Trainer and jockey overview

4.2.1 Method Explained

The betting amount reflect what the public expects from the horse. Higher the betting amount, higher the expectation of the public toward the horse. Odds are just like stock prices. Those companies which outperform the market worth more investments. For those horses which are of higher quality, gamblers are more willing to spend money on them. Not only do odds contain information about the expected performances of the horses but also the jockeys and trainers. Based on this intuition, public's expectation of the winning probability of the horse can be calculated as $\frac{1}{odds} \times 0.82$. Here, 0.82 is HKJC's commission fee. If the jockey rode 9 races in 2003, we sum up all the expected winning probability and count it as the proxy of jockey's expected number of winning races.



horse	rank	odds	public's expected winning rate
1	2	6.3	0.1317460317
2	3	2.9	0.2862068966
3	1	4.5	0.1844444444
4	5	6.7	0.123880597
5	4	5.2	0.1596153846
6	3	11	0.07545454545
7	9	15	0.05533333333
8	5	19	0.04368421053
9	2	2.8	0.2964285714

Figure 11: Public's Expectation of a Jockey

Here we construct a solid example for better illustration. By summing up the public's expected winning rate, we get 1.340447 which means the public think in these 9 races, the jockey will win an average of 1.34 races. The sample size in the example is not large enough to provide a promising result. Therefore, we collected all the races by jockey Douglas Whyte in a season and did the aforementioned analysis. The public's expectation was 103.0159 races and in the end he won 106 races.

4.2.2 Normal Distribution as a metric

We can model whether a jockey win or not in a race as a Bernoulli distribution. If the probability for the jockey to win the race is p, the variance is $p(1-p)$. We can see in figure 11, the expected winning rate of horse 1 is 0.1317. Therefore we estimate the variance as $0.1317 \times (1 - 0.1317) = 0.1144$. We sum up all the expected wins and variances as an estimate of the winning probability p. If the sample size is large, by Central Limit Theorem, the number of winning races shall follow a normal distribution. By some simple calculation, it's $N(1.34, 1.074)$ in the example. If the cumulative distribution function of this random variable is higher than 0.5, that means the jockey performs better than the public's expectation.

4.2.3 App Demo

In the App, a combination of scatter plot and bubble chart is used to visualize the public bias on performance of jockey or trainer. Figure 12 illustrates the performance of jockeys in a particular racing year. The yellow dots in the scatter plot represent the jockeys who perform better than the public's expectation. The green dots represent those jockeys who fail to meet the public's expectation. The size of the bubbles indicate absolute difference between the public's expectation and the realized number of wins.

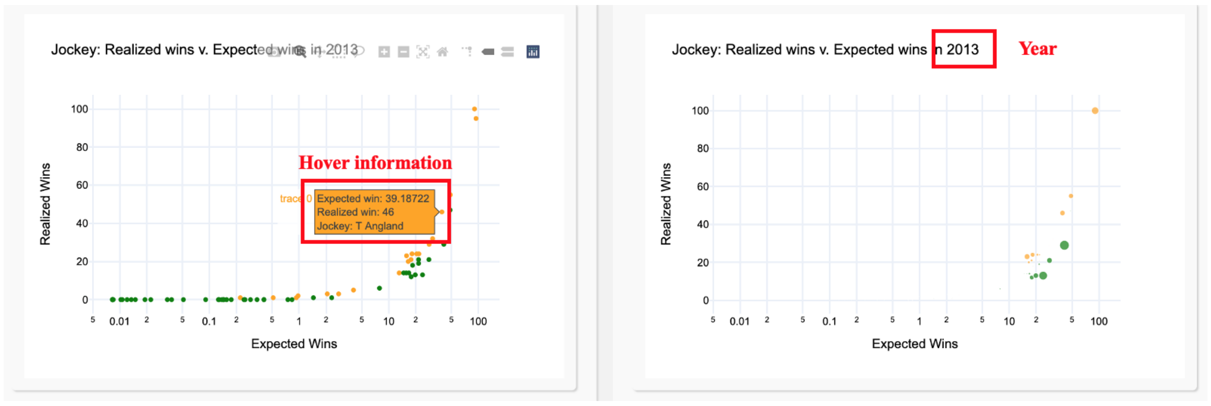


Figure 12: Performance of Jockeys

For trainers, the same methodology has been applied. Figure 13 demonstrates the performance of trainers in a particular year. Here to distinguish graphs between jockey and trainer, dots are colored in red and blue. The red dots represent outperformed cases and the blue dots represent underperformed cases. The size of the bubbles allows users to visualize public bias on each trainer.

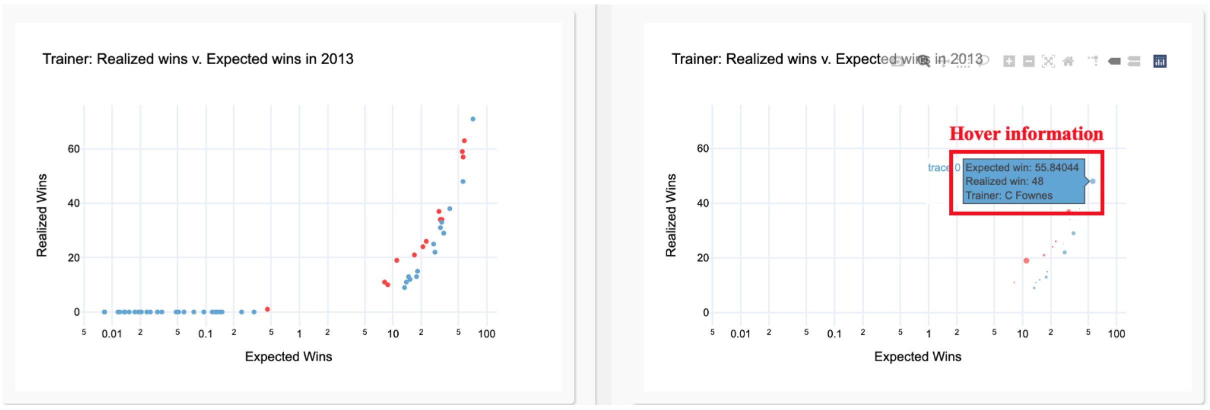


Figure 13: Performance of Trainers

For further analysis, users can move from an overview of jockey and trainer performance into deeper investigation of a single one. As shown in Figure 14 and 15, they can type in the name of particular trainer or jockey that they are interested in. A corresponding time series plot of the performance of the jockey or trainer will be displayed, which shows the trend of one's performance over year. By hovering mouse over points, users can check the winning rate i.e. the ratio of being in first three places in a certain year.

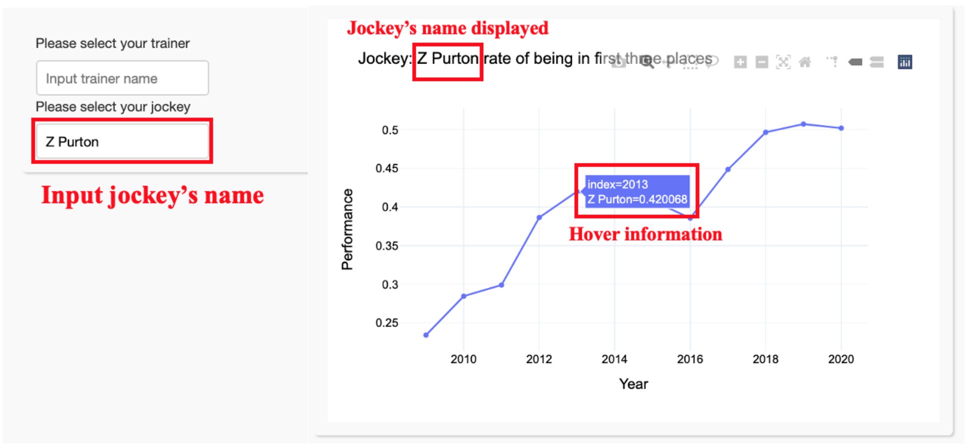


Figure 14: Performance of a particular jockey

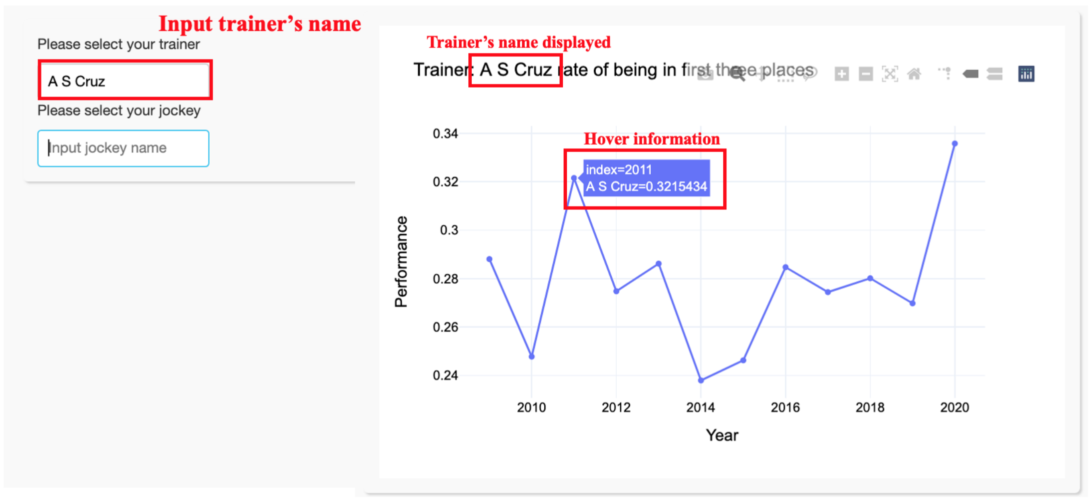


Figure 15: Performance of a particular trainer

More detailed information on the trainer will be shown by a pie chart, users can select the racecourse. Trainer's name together with his total winning rate in career life will be displayed in the title. Then the number of places and wins of this trainer in a certain racecourse (or both) can be obtained by hovering the mouse over the sectors.

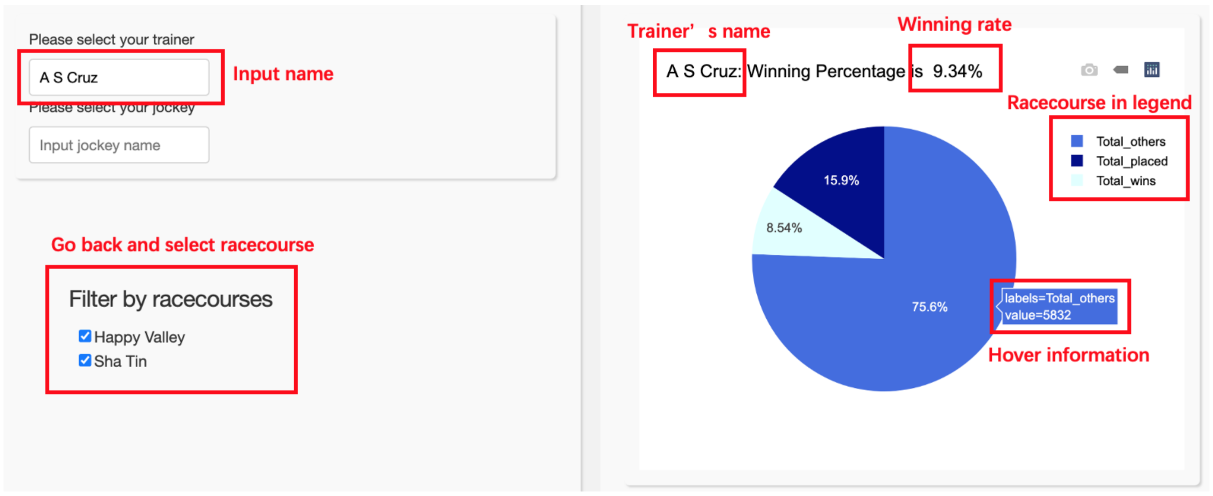


Figure 16: Single Trainer's Career Life Analysis

5 Machine Learning

The main part of the app is the model using machine learning. We have chosen XGboost and ADAboost as our final models to predict the horse racing results by comparing the results and the feasibilities of different models. The details about how we come up with the model and the model results will be presented in this section.

5.1 Model Selection

5.1.1 Multinomial Logistic Regression

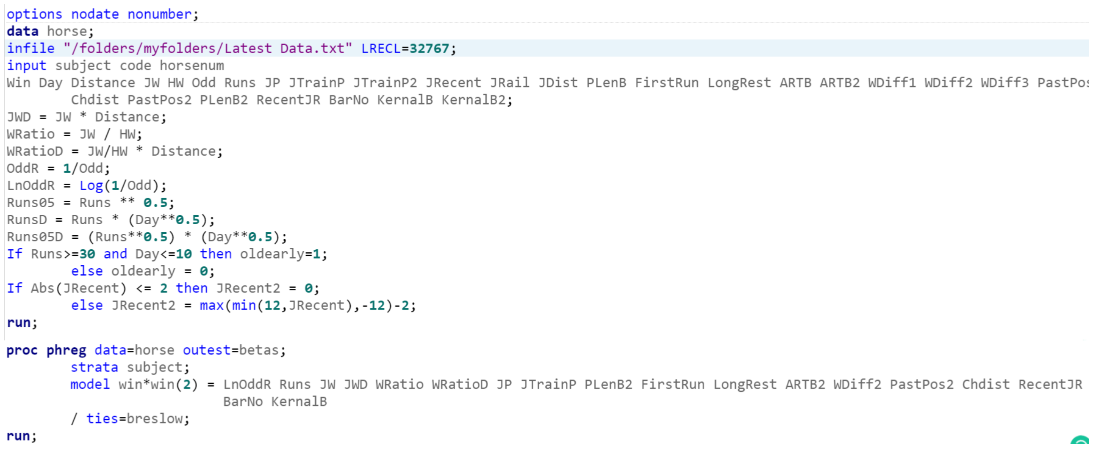
The professional gambler William Benter has succeeded in winning 1 billion dollars from the HKJC. He included his method of a long-term wagering system in his famous report. Benter (2008) claimed that he used the multinomial logistic regression to estimate the winning probability of each horse in a race. However, Benter stated that his model was presented in a detailed way in Bolton and Chapman's essay. In their essay, we have found the result. Bolton and Chapman (1986) proposed a utility model $U_h = U(\mathbf{x_h}, \mathbf{y_h})$ to measure the overall worth of a horse in a race. We can decompose the model into $U = V + \epsilon$. Since U is a utility model, the probability of horse h^* winning the race can be written as $P_h^* = P(U_h^* \geq U_h)$. Different from other linear models, they assume the error term to follow the double exponential distribution. Lau (2001) has provided the detailed derivation of the P_h^* to the closed-form expression of the probability:

$$P_h^* = \frac{exp(V_h*)}{\sum_{i=1}^{H} exp(V_h)}$$

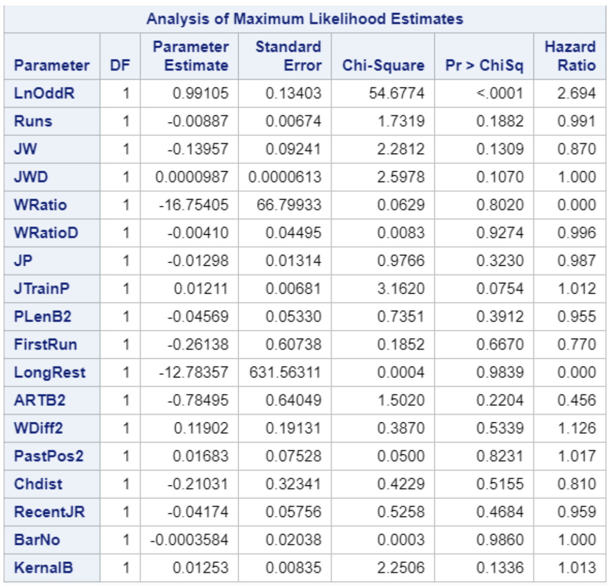
She has proposed four linear models for the deterministic function V. Her feature selection has contributed a lot to our final model:

$$
\begin{aligned}
V_{ij} = {} & \beta_1 Hwinper + \beta_2 wt.carried + \beta_3 avesprat + \beta_4 rating + \beta_5 drawing \\
& + \beta_6 Jwinper + \beta_7 Tipster + \beta_8 Tipster4 + \beta_9 Iage + \beta_{10} log.odds \\
& + \beta_{11} wt.dist + \beta_{12} age.dist
\end{aligned}
$$

Although, we did not implement the model in our app, but Wong (2011) has provided an old dataset which allows us to implement the multinomial logistic regression. Here we presents the results using SAS and Excel.

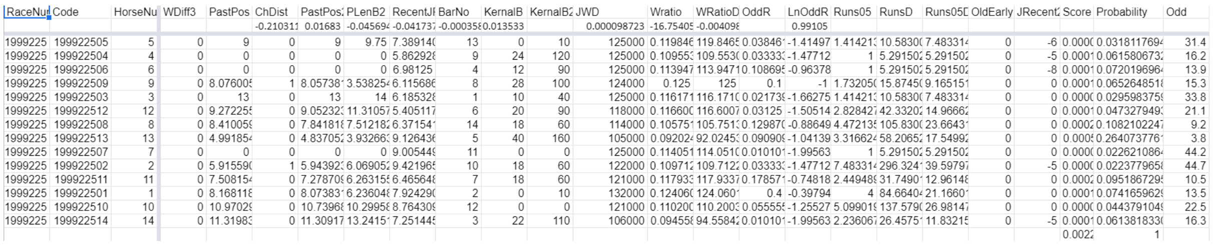


```
options nodate nonumber;
data horse;
infile "/folders/myfolders/Latest Data.txt" LRECL=32767;
input subject code horsenum
Win Day Distance JW HW Odd Runs JP JTrainP JTrainP2 JRecent JRail JDist PLenB FirstRun LongRest ARTB ARTB2 WDiff1 WDiff2 WDiff3 PastPos
        Chdist PastPos2 PLenB2 RecentJR BarNo KernalB KernalB2;
JWD = JW * Distance;
WRatio = JW / HW;
WRatioD = JW/HW * Distance;
OddR = 1/Odd;
LnOddR = Log(1/Odd);
Runs05 = Runs ** 0.5;
RunsD = Runs * (Day**0.5);
Runs05D = (Runs**0.5) * (Day**0.5);
If Runs>=30 and Day<=10 then oldearly=1;
        else oldearly = 0;
If Abs(JRecent) <= 2 then JRecent2 = 0;
        else JRecent2 = max(min(12,JRecent),-12)-2;
run;
proc phreg data=horse outest=betas;
        strata subject;
        model win*win(2) = LnOddR Runs JW JWD WRatio WRatioD JP JTrainP PLenB2 FirstRun LongRest ARTB2 WDiff2 PastPos2 Chdist RecentJR
                           BarNo KernalB
        / ties=breslow;
run;
```



Analysis of Maximum Likelihood Estimates						
Parameter	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio
LnOddR	1	0.99105	0.13403	54.6774	<.0001	2.694
Runs	1	-0.00887	0.00674	1.7319	0.1882	0.991
JW	1	-0.13957	0.09241	2.2812	0.1309	0.870
JWD	1	0.0000987	0.0000613	2.5978	0.1070	1.000
WRatio	1	-16.75405	66.79933	0.0629	0.8020	0.000
WRatioD	1	-0.00410	0.04495	0.0083	0.9274	0.996
JP	1	-0.01298	0.01314	0.9766	0.3230	0.987
JTrainP	1	0.01211	0.00681	3.1620	0.0754	1.012
PLenB2	1	-0.04569	0.05330	0.7351	0.3912	0.955
FirstRun	1	-0.26138	0.60738	0.1852	0.6670	0.770
LongRest	1	-12.78357	631.56311	0.0004	0.9839	0.000
ARTB2	1	-0.78495	0.64049	1.5020	0.2204	0.456
WDiff2	1	0.11902	0.19131	0.3870	0.5339	1.126
PastPos2	1	0.01683	0.07528	0.0500	0.8231	1.017
Chdist	1	-0.21031	0.32341	0.4229	0.5155	0.810
RecentJR	1	-0.04174	0.05756	0.5258	0.4684	0.959
BarNo	1	-0.0003584	0.02038	0.0003	0.9860	1.000
KernalB	1	0.01253	0.00835	2.2506	0.1336	1.013

Figure 17: SAS code and the results of the model



RaceNu	Code	HorseNu	WDiff3	PastPos	ChDist	PastPos2	PLenB2	RecentJ	BarNo	KernalB	KernalB2	JWD	Wratio	WRatioD	OddR	LnOddR	Runs05	RunsD	Runs05D	OldEarly	JRecent2	Score	Probability	Odd
					-0.210311	0.01683	-0.045694	-0.041737	-0.00035	0.013533		0.000098723	-16.75405	-0.004098		0.99105								
1999225	199922505	5	0	9	0	9	9.75	7.38914	13	0	10	125000	0.11984	119.8465	0.03846	-1.41497	1.41421	10.5830	7.48331	0	-6	0.0000	0.031811769	31.4
1999225	199922504	4	0	0	0	0	0	5.86292	9	24	120	125000	0.10955	109.553	0.03333	-1.47712	1	5.29150	5.29150	0	-5	0.0001	0.061580673	16.2
1999225	199922506	6	0	0	0	0	0	6.98125	4	12	90	125000	0.113947	113.9471	0.10869	-0.96378	1	5.29150	5.29150	0	-8	0.0001	0.072019696	13.9
1999225	199922509	9	0	8.07600	1	8.05738	3.53825	6.11568	8	28	100	124000	0.125	125	0.1	-1	1.73205	15.8745	9.165151	0	0	0.0001	0.065264851	15.3
1999225	199922503	3	0	13	0	13	14	6.18532	1	10	40	125000	0.116171	116.171	0.02173	-1.66275	1.41421	10.5830	7.48331	0	0	0.0000	0.029598375	33.8
1999225	199922512	12	0	9.27225	0	9.05232	11.31057	5.405117	6	20	90	118000	0.11660	116.6007	0.03125	-1.50514	2.828427	42.3320	14.9666	0	0	0.0001	0.047327949	21.1
1999225	199922508	8	0	8.41005	0	7.84181	7.51218	6.371541	14	18	60	114000	0.105751	105.751	0.12987	-0.88649	4.47213	105.830	23.6643	0	0	0.0002	0.108210224	9.2
1999225	199922513	13	0	4.99185	0	4.83705	3.93266	9.12643	5	40	160	105000	0.09202	92.0245	0.09090	-1.04139	3.316624	58.2065	17.5499	0	0	0.0005	0.26407377	3.8
1999225	199922507	7	0	0	0	0	0	9.00544	11	0	0	125000	0.114051	114.051	0.010101	-1.99563	1	5.29150	5.29150	0	0	0.0000	0.022621086	44.2
1999225	199922502	2	0	5.91559	1	5.94392	6.06905	9.42196	10	18	60	122000	0.10971	109.712	0.03333	-1.47712	7.48331	296.324	39.5979	0	-5	0.0000	0.022377965	44.7
1999225	199922511	11	0	7.50815	0	7.27870	6.26315	6.46564	7	18	60	121000	0.11793	117.9337	0.178571	-0.74818	2.44948	31.7490	12.9614	0	0	0.0002	0.095186729	10.5
1999225	199922501	1	0	8.16811	0	8.07383	6.23604	7.92429	2	0	10	132000	0.12406	124.060	0.4	-0.39794	4	84.6640	21.1660	0	0	0.0001	0.074165962	13.5
1999225	199922510	10	0	10.9702	0	10.7396	10.2995	8.76430	12	0	0	121000	0.11020	110.200	0.05555	-1.25527	5.09901	137.579	26.9814	0	0	0.0000	0.044379104	22.5
1999225	199922514	14	0	11.3198	0	11.30917	13.2415	7.25144	3	22	110	106000	0.09455	94.5584	0.010101	-1.99563	2.236067	26.4575	11.8321	0	-5	0.0001	0.061381833	16.3
																						0.0022	1	

Based on the parameter estimates and the logit function, we can calculate the winning probability of each horse in a race as represented in the last two columns of the table. Wong (2011) claimed that once the model had an accuracy over 18%, it's possible to form a sure win strategy. Furthermore, professional gambling teams apply multinomial logistic regression into actual horse racing predictions. However, the result was stated about decades ago. In order to beat the market nowadays, the model has to be even more accurate, but we fail to get a similar prediction accuracy due to a lack of features.

5.1.2 Boosting Algorithms

Boosting algorithms are ensemble learning methods that learn from the errors made by weak classifiers and correct the descrepancies in the final results. Schapire (1990) mentioned that the idea of boosting algorithms was introduced when Kearns and Valiant proposed if weak models could be combined into a stronger one. When solving real-world problems, with significant degrees of hardness and sensitivee to specific time being, by machine learning, the predictions by weak models after observations can be optimised with boosting algorithms. The boosting algorithms suit for meeting the objectives of more accurate predictions of horse racing results given the extent of multi-dimensional variables and the ultra dynamic racing environments.

The study used two boosting algorithms, AdaBoost and eXtreme gradient boosting (XGBoost) using Decision Tree or Linear Regression as its booster, which were trained by the relevant horse racing dataset collected from 2017-2019, validated afterwards by the entries in 2020. The models were deployed to predict ongoing races. In each of the models, we made adjustments to hyperparameters. After some training processes, we chose a learning rate of 0.1 as an entrant of gradient descent, a max-depth of 6 for weak tree classifiers to prevent over-fitting, and 100 estimators for better prediction performances.

AdaBoost Almost a decade after Kearns and Valiant outlined the blueprint of boosting algorithms, Freund et al. (1996) developed Adaptive Boosting (AdaBoost), a boosting algorithm to enhance the performance of machine learning models, especially weak classifiers which are slightly better than random guessing. AdaBoost has a great learning capability with efficiency and simplicity which elevates accurate pattern identifications.

Schapire (1999) explained the algorithm of AdaBoost. The final hypothesis of AdaBoost is:

$$H(x) = sign(\sum_{t=1}^{T} \theta_t f_t(x))$$

Where t represents a sequence of rounds of invoking weak models, θ_t measures the importance (weights) of a weak classifier in a specific prediction, and $f_t(x)$ measures how confident a prediction is.

In this research, the AdaBoost model used Random Forest as its base estimator. Lv et al. (2019) mentioned that Random Forest with AdaBoost can improve the performances of inferences in binary classifications.

XGBoost Friedman et al. (2000) introduced the framework of XGBoost as a boosting algorithm within a wide range of applications in real-world domains like AdaBoost, and it also supports to solve ranking and classification problems. Dhaliwal et al. (2018) mentioned that XGBoost was the key to the winners in Kaggle machine learning competition in 2015 because of its capabilities in parallel processiong, regularizations, and flexibility. Besides, Chen et al. (2015) also mentioned XGBoost's incredible preformance in predictions involving various datasets, which is a good-fit for training set of horse races combining the data of jockey, trainer, and horses.

5.2 Model Performance

Confusion matrices and ROC Curves were obtained with the prepared testing set and summarised the to evaluate the models.

Confusion Matrices As we labeled horses ranked first-three as 1 (Positive) or otherwise as 0 (Negative), the i-th row and j-th column respectively represent the true classes and predicted classes of horses.

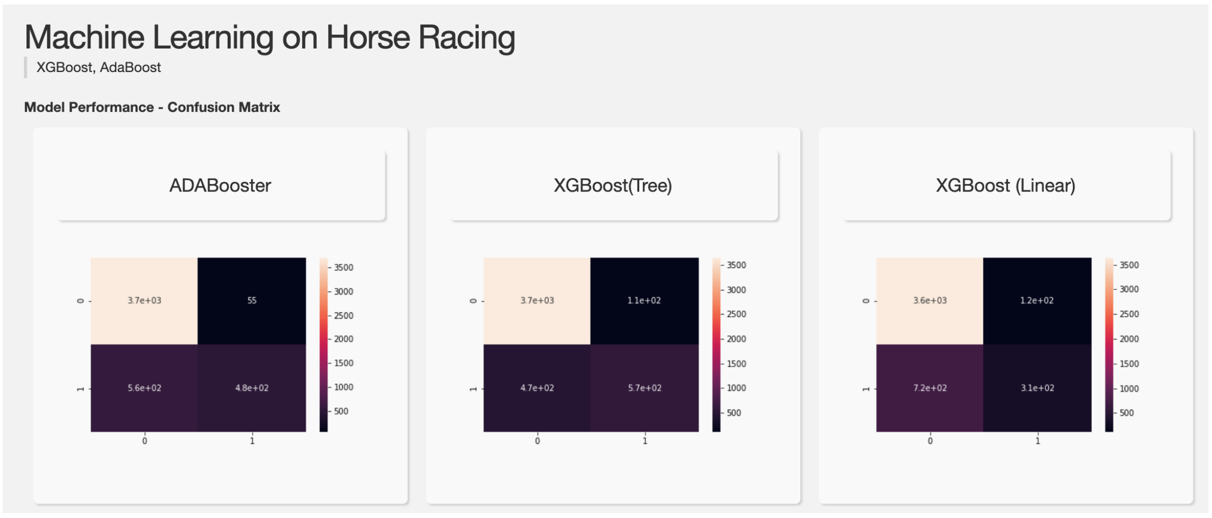
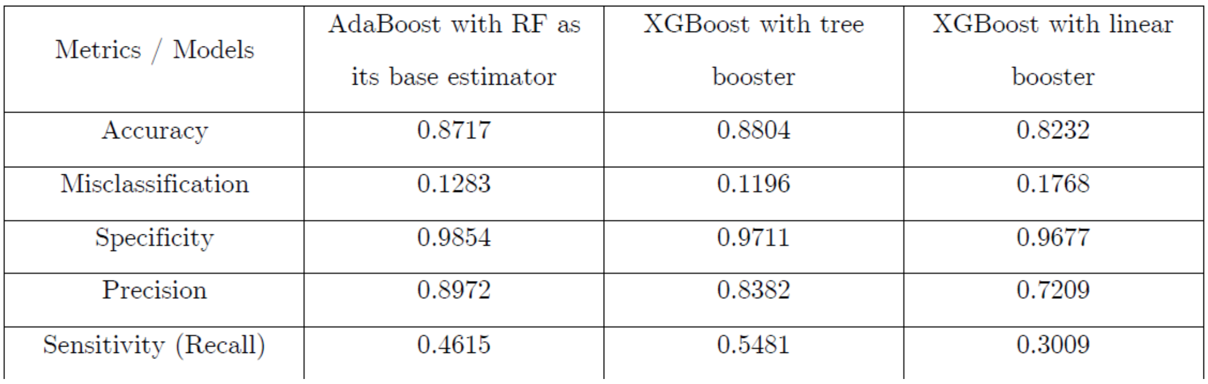


Figure 18: Confusion Matrices

The subgrids of each of the matrices include True Negative (TN), True Positive (TP), False Positive (FP or Type 1 error), and False Negative (FN or Type 2 error). We calculated five different metrics for model evaluations, including accuracy, misclassification, specificity, sensitivity, and precision, from the matrices.



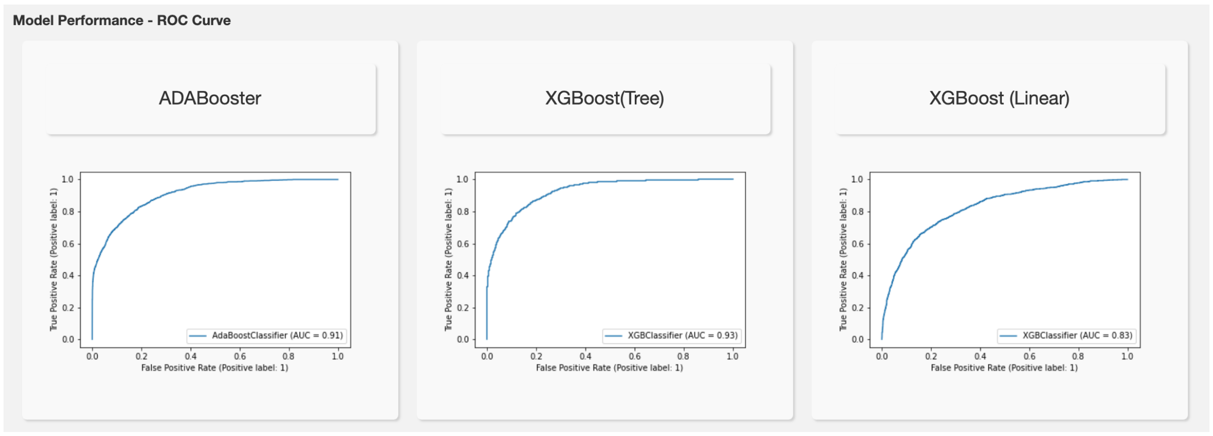
Metrics / Models	AdaBoost with RF as its base estimator	XGBoost with tree booster	XGBoost with linear booster
Accuracy	0.8717	0.8804	0.8232
Misclassification	0.1283	0.1196	0.1768
Specificity	0.9854	0.9711	0.9677
Precision	0.8972	0.8382	0.7209
Sensitivity (Recall)	0.4615	0.5481	0.3009

Figure 19: Summarise metrics of Confusion Matrices

XGBoost with tree booster scored the best in term of its capability in recognising the patterns in the testing set with minimum misclassification. All three models score higher than 0.96 in specificity, which means that they are good in finding out horses which are not first-three, and such results are intuitive as it's relatively easier to predict not first-three predict horses.

Speaking about Precision, AdaBoost is the greatest among the three as the users of models can be most confident in trusting the predicted first-three horses. On the other hand, the results of sensitivity tell us that XGBoost with tree booster can identify the largest proportion of positives in actual positives.

ROC Curves We obtained ROC curves for three models, and their Area Under Curve (AUC) scores will be used to evaluate their performance, with x-axis representing False Positive Rate and y-axis representing True Positive Rate.

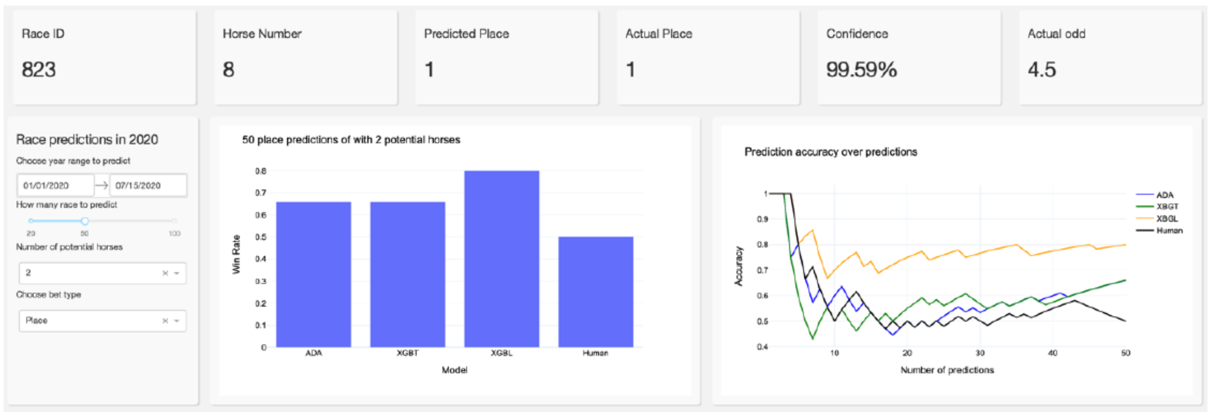


	AdaBoost with RF as its base estimator	XGBoost with tree booster	XGBoost with linear booster
AUC Score	0.91	0.93	0.83

Figure 20: ROC curves and AUC Scores

From the results of AUC scores, XGBoost with tree booster has the highest score compared to other 2 models. Thus, we can tell that XGBoost with tree booster have a better performance in identifying first-three horses in the testing set.

5.3 Predicting Horse Races on Web Application



Race ID: 823
Horse Number: 8
Predicted Place: 1
Actual Place: 1
Confidence: 99.59%
Actual odd: 4.5

Race predictions in 2020
Choose year range to predict: 01/01/2020 → 07/15/2020
How many race to predict: 50
Number of potential horses: 2
Choose bet type: Place

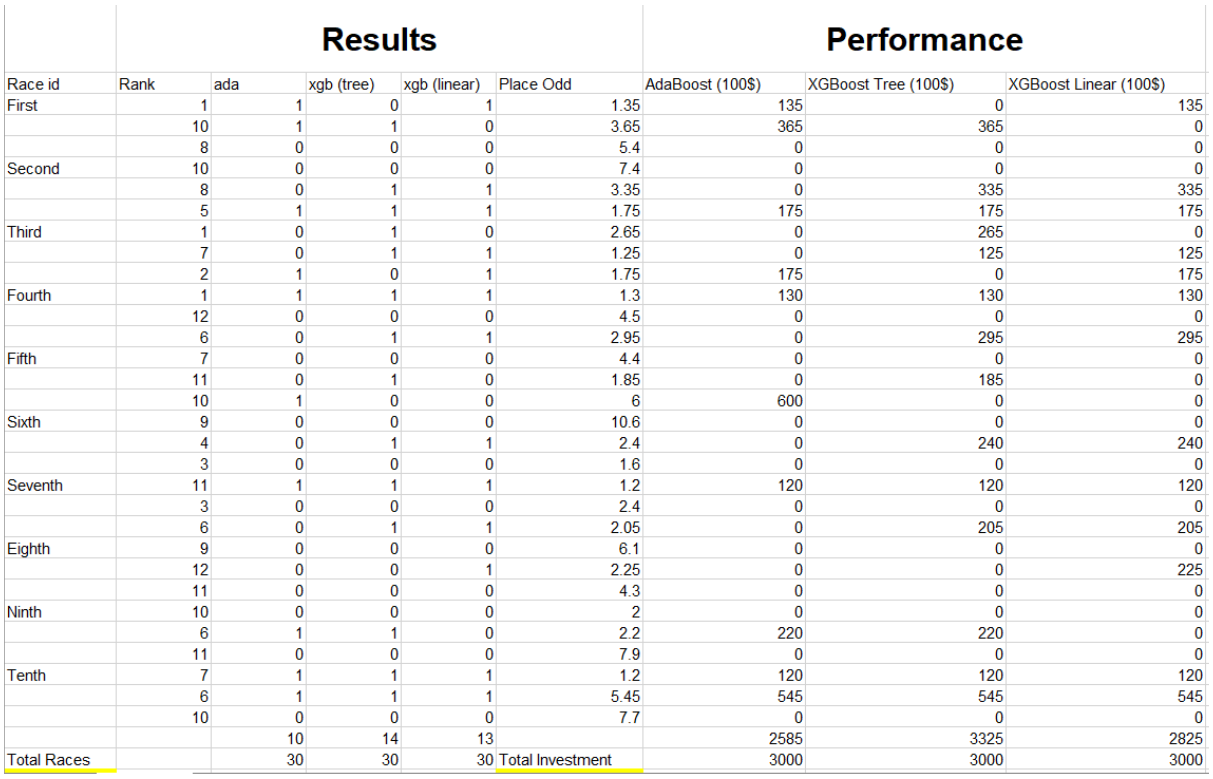
Figure 21: The Prediction Interface

On our application, the users could use our models to see the accuracy of predictions in different bet types in 2 main steps: Firstly, the users can choose the interested time period in 2020 and the number of races for predictions predict. Secondly, the users could specify the number of potential horses (aka horses with top predicted winning rates) and the betting type. The real-time predictions information such as the confidence of predictions and actual odds are provided.

The results of predictions on the web application are illustrated using the bar chart and line plot for accuracy visualisations among 3 models and their comparisons with simply betting by odds (Human).

6 Real Application

Although the validation sets' results are satisfactory initially, the accuracy of the model in real-life prediction is still doubted. Therefore, we deployed the models to assist real betting of horse racing games at 9th of May in 2021. The performances of the models are evaluated in terms of the return in betting the type Place. If the gambler's chosen horse has successfully entered the top 3, the gambler will be awarded. The results of the races and the predictions of the models are organized in the table below:



	Results					Performance		
Race id	Rank	ada	xgb (tree)	xgb (linear)	Place Odd	AdaBoost (100$)	XGBoost Tree (100$)	XGBoost Linear (100$)
First	1	1	0	1	1.35	135	0	135
	10	1	1	0	3.65	365	365	0
	8	0	0	0	5.4	0	0	0
Second	10	0	0	0	7.4	0	0	0
	8	0	1	1	3.35	0	335	335
	5	1	1	1	1.75	175	175	175
Third	1	0	1	0	2.65	0	265	0
	7	0	1	1	1.25	0	125	125
	2	1	0	1	1.75	175	0	175
Fourth	1	1	1	1	1.3	130	130	130
	12	0	0	0	4.5	0	0	0
	6	0	1	1	2.95	0	295	295
Fifth	7	0	0	0	4.4	0	0	0
	11	0	1	0	1.85	0	185	0
	10	1	0	0	6	600	0	0
Sixth	9	0	0	0	10.6	0	0	0
	4	0	1	1	2.4	0	240	240
	3	0	0	0	1.6	0	0	0
Seventh	11	1	1	1	1.2	120	120	120
	3	0	0	0	2.4	0	0	0
	6	0	1	1	2.05	0	205	205
Eighth	9	0	0	0	6.1	0	0	0
	12	0	0	1	2.25	0	0	225
	11	0	0	0	4.3	0	0	0
Ninth	10	0	0	0	2	0	0	0
	6	1	1	0	2.2	220	220	0
	11	0	0	0	7.9	0	0	0
Tenth	7	1	1	1	1.2	120	120	120
	6	1	1	1	5.45	545	545	545
	10	0	0	0	7.7	0	0	0
		10	14	13		2585	3325	2825
Total Races		30	30	30	Total Investment	3000	3000	3000

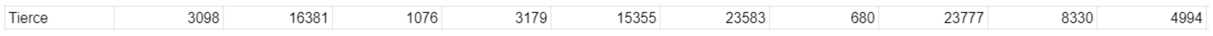
Figure 22: Result of the 10 races

The models' prediction results are unsatisfactory for the ten races. All of the three models attain a low accuracy in predicting the winner. Luckily, the horse that the model claimed to be the winner had a high probability of entering the top 3. If we choose the place as our betting type and strictly follow the result of the models, the return can be positive or negative depending on the model. For AdaBoost, the return was -425 HKD; for XGBoost Tree, the return was 325HKD; and lastly, the return of XGBoost Linear was -185 HKD.

6.1 Alternative betting strategy

Although the rules of win and place are easy, the odds and the low prediction probability make it very difficult to earn a profit. Tierce is a type of bet that allows the gambler to predict the order of the top 3 horses where we spot some quick cash. For example, if the real result is horse number 2,3,4 getting the first, second and third place. By submitting the number 2,3,4 accordingly, you will win the bet. Tierce has a considerable return with a cost of 10 HKD per bet.

The strategy is illustrated here. Firstly, we can figure out the top 2 horses with the highest winning probabilities by using the model. Secondly, we get rid of the two best horses and the two worst horses with the highest odds and randomly select four more horses. We buy all the permutations of the six horses meaning that we will place a total number of $P_3^6 = 120$ bets. The cost of each wager is 10 HKD; therefore, the cost of each race is 1200 HKD. The only way to earn a return is to predict the underdog or sleepers correctly. The regular return of tierce can be several hundred only, but the abnormal return with underdogs can be more than 10K! The table below shows the profit of tierce of the ten races at 9th of May. The intuition of the strategy is to seize the high odds of tierce and attempt to earn the underdog's money. Notice that due to technical difficulties like trading 120 bets for each race, an inadequate amount of initial capital, we fail to apply the strategy in real life. But using the ten races as a simulation, our ADA booster model has successfully predicted the tierce of the second race and the fourth race, meaning a total return of 8000 HKD. Once the abnormal odds are captured, all the costs are eliminated.



Tierce	3098	16381	1076	3179	15355	23583	680	23777	8330	4994

Figure 23: Dividends of Tierce for the 10 races

7 Reflection

As cited in Wong (2011), no one has succeeded in predicting the winner by purely using statistical models in reality. In our real application, we find it very hard to fulfill the data requirement of our model. For instance, some new trainers and horses participating in a race do not have any previous records. It adds a tremendous amount of bias and variance to our model prediction when applying it. A combination of a new horse and an experienced jockey like Z Purton might have extraordinary performance in reality. The model fails to predict this kind of situation. However, with the assistance of the recency weighted rank of other horses, with the traditional jockey analysis, users still have a way to spot these horses out.

After the first actual application, the models were refined according to errors in the predictions:

Firstly, we spotted that the training set was not well standardized and normalized as seen in imbalanced feature importance. Thus, we prepared another collection of improved preprocessed data for models to be re-trained.

Furthermore, we noticed that we mistakenly shuffled the entries of the training set in which the data did not follow the time series. As a result, it led to overfitting and distorted weights of attributes. The time series problem shall explain why the models can achieve such a higher accuracy in the presentation. Thus, we rearranged rows in training and validation datasets by sorted dates in ascending order for re-training. Furthermore, we dealt with the problem by using the adaptive scheme that we start with only a few races and gradually including the later races to avoid the overfitting problem. The re-trained model did not have future data as assistance. Therefore, the accuracy dropped to 0.4.

In the future timeframe, we plan to keep refining our models as time progresses. For instance, we would like to increase the number of features by including more attributes of jockeys and trainers. In order to boost the model's performance, a real-time updated dataset has to be implemented into the model.

8 Conclusion

In conclusion, the prediction accuracy of the overall model is still satisfactory, but horse racing can not be fully determined by statistical learning. It is quite crucial to follow human instinct and traditional analysis once you suspect some potential errors in the model. However, as a man who wants to earn a profit in the race, the model and the app definitely assist you in making bright decisions.

References

Benter, W. (2008). Computer based horse race handicapping and wagering systems: a report. In *Efficiency of racetrack betting markets*, pp. 183–198. World Scientific.

Bolton, R. N. and R. G. Chapman (1986). Searching for positive returns at the track: A multinomial logit model for handicapping horse races. *Management Science 32*(8), 1040–1060.

Chen, T., T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, et al. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2 1*(4).

Dhaliwal, S. S., A.-A. Nahid, and R. Abbas (2018). Effective intrusion detection system using xgboost. *Information 9*(7), 149.

Freund, Y., R. E. Schapire, et al. (1996). Experiments with a new boosting algorithm. In *icml*, Volume 96, pp. 148–156. Citeseer.

Friedman, J., T. Hastie, R. Tibshirani, et al. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Annals of statistics 28*(2), 337–407.

Lau, S. (2001). *Some Statistical Analysis of Handicap Horse Racing.* Ph. D. thesis, The Chinese University of Hong Kong.

Lv, Y., X. Shi, L. Ran, and M. Shang (2019). Random forest-based ensemble estimator for concrete compressive strength prediction via adaboost method. In *The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, pp. 557–565. Springer.

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning 5*(2), 197–227.

Schapire, R. E. (1999). A brief introduction to boosting. In *Ijcai*, Volume 99, pp. 1401–1406. Citeseer.

Wong, C. (2011). *Precision: Statistical and mathematical methods in horse racing.* Outskirts Press, Incorporated.