Predicting Horse Racing Result using Tensorflow

LYU1603 Final Year Project Term 2 Report

LAU Ming Hei (1155051346)

Supervised by Prof. LYU Ring Tsong Michael

Table of Contents

Abstract

Estimating horse racing result has been a popular topic in machine learning field, whilst the possibility of profit earning depends on the accuracy of predicting the probabilities of horses to win in a race. Due to the comprehensive historical data provided by the Hong Kong Jockey Club, a lot of experiments could be done. This report would describe a new approach to normalize data, to train and evaluate model on Tensorflow and to compare the new model with the model trained in last semester. We show that a trained linear regression model performed better on ranking horses in a race, and a trained linear classification model, which is the model trained last semester, performed better on playing win bet.

Introduction

Motivation

Horse racing has been a famous topic in machine learning field, while the recent performance of deep neural network is stunning and there were a lot of new machine learning tools released recently, which could let us apply deep learning algorithm or other machine learning algorithm easily, so that we would like to conduct an experiment on predicting horse racing result.

Background

Horse Racing

Horse racing is a sport that running horses at speed. In Hong Kong, horse racing is not purely a sport, it has gambling components associated. Around 8-14 horses in a race, these are only one type of race in Hong Kong, the faster the winner. However, there are different types of betting, such as win bet, which is guessing the winner; Jockey Challenge, which is the best performance jockey.

Hong Kong Jockey Club

"The Hong Kong Jockey Club (HKJC) is a non-profit organization providing horse racing, sporting and betting entertainment in Hong Kong. It holds a government-granted monopoly in providing pari-mutuel betting on horse racing. The organization is the largest taxpayer in Hong Kong, as well as the largest community benefactor."[1]

Pari-mutuel betting

"Pari-mutuel betting is a betting system in which all bets of a particular type are placed together in a pool and taxes are removed, and payoff odds are calculated by sharing the pool among all winning bets."[2]

"Dividend will be shared by the number of winning combinations of a particular pool. Winners will share the Net Pool in proportion to their winning stakes."[3]

[1] https://en.wikipedia.org/wiki/Hong_Kong_Jockey_Club
[2] https://en.wikipedia.org/wiki/Parimutuel_betting
[3] http://special.hkjc.com/racing/info/en/betting/guide_qualifications_pari.asp

Types of bets

In Hong Kong, HKJ provided a wide range of bets and each type of bet has its own pool.

Name	Description[4]	Betting system
Win	1st in a race	Pari-mutuel
Place	1st, 2nd or 3rd in a race with 7 or more declared starters or 1st, 2nd in a race with 4, 5, 6 declared starters	Pari-mutuel
Quinella	1st and 2nd in either order in the race	Pari-mutuel
Quinella Place	Any two of the first three placed horses in any finishing order in the race	Pari-mutuel
Trio	1st, 2nd and 3rd in any order in the race	Pari-mutuel
Tierce	1st, 2nd and 3rd in correct order in the race	Pari-mutuel
First Four	1st, 2nd, 3rd and 4th in any order in the race. (Merged pool with Quartet)	Pari-mutuel
Quartet	1st, 2nd, 3rd and 4th in correct order in the race. (Merged pool with First Four)	Pari-mutuel
Double	1st in two nominated races – 1st in 1st leg and 2nd in 2nd leg	Pari-mutuel

[4] https://en.wikipedia.org/wiki/Hong_Kong_Jockey_Club

Treble	1st in three nominated races – 1st in first two legs and 2nd in third leg	Pari-mutuel
Double Trio	1st, 2nd and 3rd in any order in both legs	Pari-mutuel
Triple Trio	1st, 2nd and 3rd in any order in three legs – 1st, 2nd and 3rd in the first two Triple Trio legs	Pari-mutuel
Six Up	1st or 2nd in each of the legs nominated to comprise the Six Up – 1st or 2nd in each of the legs nominated to comprise the Six Up	Pari-mutuel
Jockey Challenge	Best performing jockey in a race meeting	Fixed-odds

Objective

Our objective is to create a model which could predict the finishing time of a horse in a race so that we could rank them accordingly and compare the new model with the old one to see which one is better.

Data Collection

Fast Approach

There exist companies offer the sale of horse racing historical data in Hong Kong, a fast approach is to buy data, though the price is considerable, due to a lack of budget, this approach is not suitable.

Web Crawling

Tailor-made python scripts were created to crawl data from the HKJC website, historical data and horses' information from 2001 to 2015 horse seasons were collected. Data were structured in csv format and there are 20 features in total. The following table is describing the structure of a row record in a race.

Feature	**Description**
Date	-
Location	-
Race Number	-
Class	-
Distance	-
Going	Track condition
Course	Track
Pool	Prize pool
Place	-
Horse ID	-
Horse	-

Jockey	-
Trainer	-
Actual Weight	Carried weight
Declare Weight	Overall weight
Draw	-
LBW	Length behind winner
Running Position	-
Time	Finishing time
Win Odds	Closing odds

The following table is describing the structure of a horse information.

Feature	**Description**
Name	The name of the horse
ID	-
Country of Origin	-
Color	-
Sex	-
Import Type	-
Sire	-
Dam	-
Dam's Sire	-

ELO Rating System

As jockey, horse and trainer itself cannot be used as a single feature, we should find a way or a value to represent their existence, and we have chosen to use ELO rating system. ELO rating system is designed for two players originally; a generalized formula is required to apply it on horse racing.

ELO Rating System for multiple player[5]

A generalized ELO rating system for multiple player has been found on the internet[6], then we have slightly modified it to fit the horse racing sanatoria.

Estimation score function

Suppose there are N horses in a race with rating $R_1, R_2, R_3, \dots, R_N$, the estimation score for a horse x to win the game is:

$$E_x = \frac{\sum_{1<i<N, i\neq x} \frac{1}{1+10^{R_i - R_x}}}{\frac{N(N-1)}{2}}$$

Also, notice that,

$$\sum_{i=1}^{N} E_i = 1$$

[5] https://en.wikipedia.org/wiki/Elo_rating_system

[6] http://sradack.blogspot.hk/2008/06/elo-rating-system-multiple-players.html

Scoring function

The scoring function is a function of p, where p is the result of a horse, i.e. 1 place, 2 place, 3 place, …

$$S_p = \frac{N-p}{\frac{N(N-1)}{2}} \; for\ 1 \leq p \leq 14$$

Due to the following condition,

$$\sum_{i=1}^{N} S_p = 1$$

If the result place is a double head of something, which means two horses has the same finishing time, i.e. two horses both are 1 place, then their result place should be 1 place double head. For double head results, we should add 0.5 to the place,

$$S_p = \frac{N-(p+0.5)}{\frac{N(N-1)}{2}}, for\ p\ is\ double\ head, 1 \leq p \leq 13$$

Ranking Function

That the new rank of a horse after a race is,

$$R'_x = R_x + K(S_x - E_x)$$

Compute the ELO

Our goal is to compute the ELO for jockeys, horses and trainers, so that we can have a value to represent them. In the equations mentioned above, there is a variable K.

Parameter Tuning

K is a factor that scales the magnitude of the change to a player's rating after a given game[7]. We have to find a good *K*, before that we have to had a method to evaluate the *K* we have chosen is good or not. The method we proposed is the following:

[7] http://sradack.blogspot.hk/2008/06/elo-rating-system-multiple-players.html

For every race over the past 15 years, bet on the highest ELO horse/jockey/trainer, depends on what ELO we were calculating, then use the winning percentage over the past 15 years to determine the *K* is good or not.

Result

We have set the initial ELO to 1500, then we followed the equations above to compute ELO for jockeys, horses and trainers at different time points, we first pick a random K value, then do the binary search manually, there were 11074 race in total over the past 15 years, the K values we have found for horse/jockey/trainer could produce a good winning percentage, the following are the result.

Possible ways to model the problem

There are many ways to model the problem, the following are the ways we have considered.

Strength of a horse

We could train a regression model to estimate the strength of a horse, in a race, we could then bet on the horse with highest estimation score among horses. However, we don't have a numerical value that could represent the strength of a horse for us to use as the training label, so that this method is not suitable for us.

Probability of a horse to win the race

We could train a classification model to classify a horse will win the race or not. As a classification model, would produce the probability of different classes, we could then bet on the horse with the highest probability to win the race among horses. Before we can build the model, we should create a label for each record, if the place of a record is '1' or '1 DH', then we will set the label to 1, otherwise 0.

Finishing Time

We could train a regression model to estimate the finishing time of a horse in the race, in a race, we could then bet on the horse with the fastest estimated finishing time among horses.

Winner

We could train a classification model which accepts 8-14 horses' information, then classify which horse is more likely to win the race. Before we can build the model, we have to put all horses' information in a race into a vector, there are around 22 features for a horse, in the worst case, there could be 308 features in a vector.

Standardization

Intuition

The normalization or standardization method of the entire dataset, which has been used in the last semester, has been revisited. A critical problem has been found, which is normalizing or standardizing data by column primarily is not a good approach. It is because *the meaning of a value would be various depends upon the situation*. For instance, horse A finished a class-2 1500-meters race in 1 minute 20 seconds and horse B finished a class-3 1300-meters race in 1 minute 20 seconds; the performance of two horses were the same if we are considering the finishing time only; however, 1 minute 20 seconds in a class-2 1500-meters race might indicate that the performance of the horse was bad; on the contrary, 1 minute 20 seconds in a class-3 1300-meters race might indicate that the performance of the horse was good. Therefore, instead of standardize a column globally, which means calculate the mean and the standard deviation of an entire column, *we should standardize a column/value locally, which means calculate the mean and the standard deviation of the value with respect to the situation*. Using the above example, we should standardize the finishing time of horse A with the mean and the standard deviation of the finishing time of class-2 1500-meters race, and standardize the finishing time of horse B with the mean and the standard deviation of the finishing time of class-3 1300-meters race.

Figure 1 – 4 show the finishing time distributions of 1200-meters race among classes. The x-axis is finishing time in milliseconds. The y-axis is the probabilities of the finishing time. In figure 1, it shows that the finishing time distribution of class-1 and class-2 had the similar variance, and class-2 had a higher mean than class-1. Figure 2 – 4 show the similar result among remaining classes. The assumption we made for finishing time, "*the meaning of a value would be various depends upon the situation*", has been proven by figure 1 – 4. Generally speaking, *the higher the class of a race, the faster the time need to be achieved in order to be recognized as a good performance (Figure 5).*

Not only for finishing time, almost all numerical features we are going to use, which are including actual weight (Figure 6) and declare weight (Figure 8), have different distribution with respect to different situations (combination of classes and distance), except for win odd

(Figure 7), which has similar mean and variance among different classes. By looking at finishing time distributions among all situations (Figure 9), we can agree that a finishing time value has different meaning depends on the type of class and distance, and we should standardize the finishing time according to its situation.

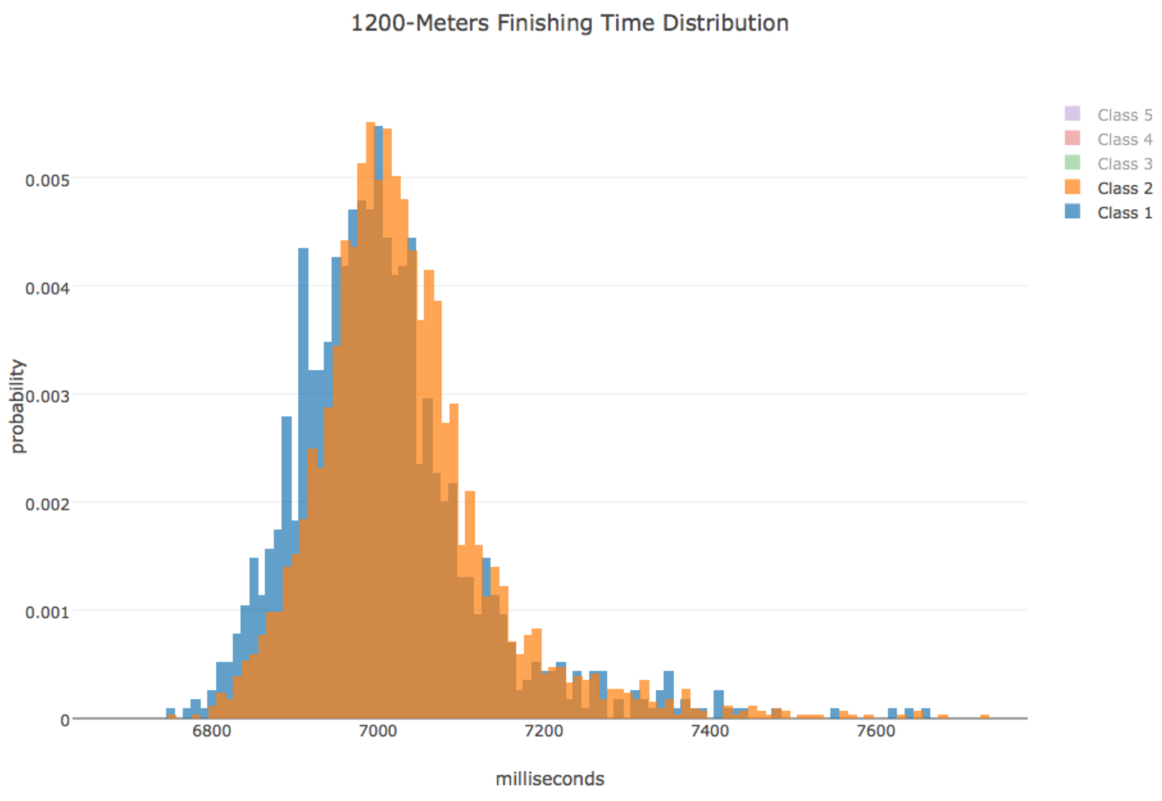


Figure 1

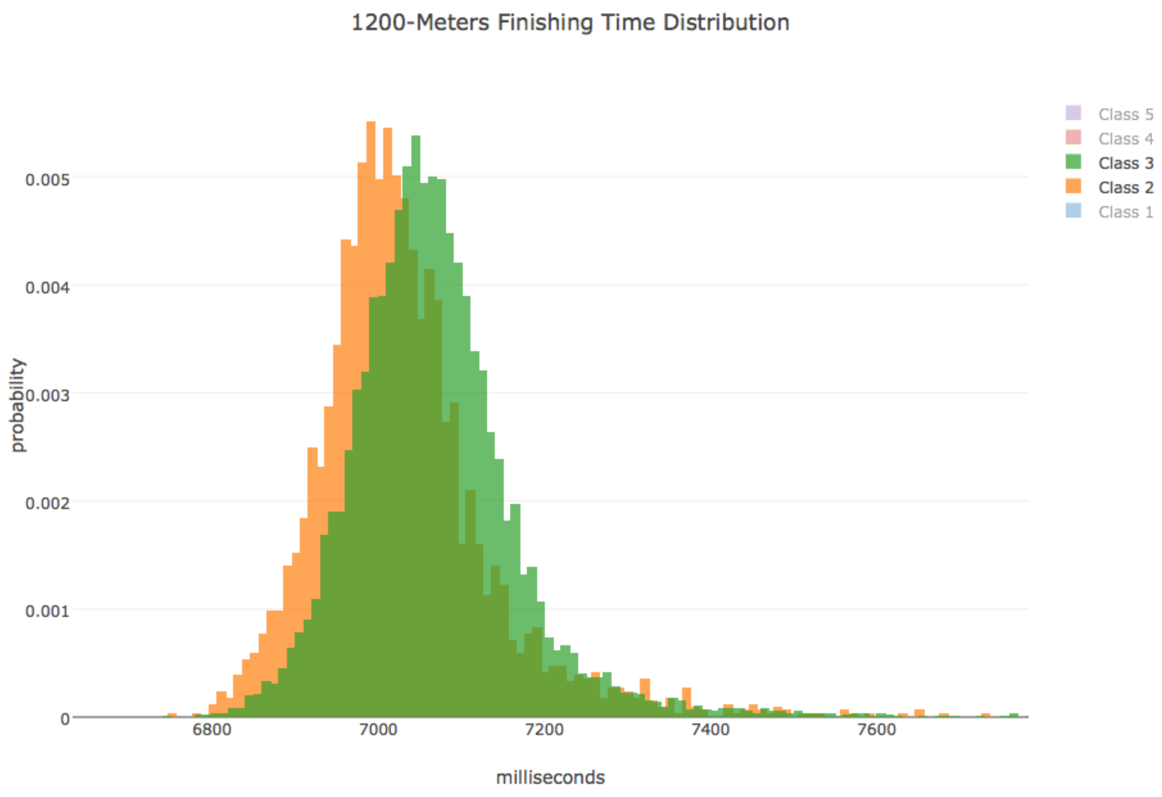


Figure 2

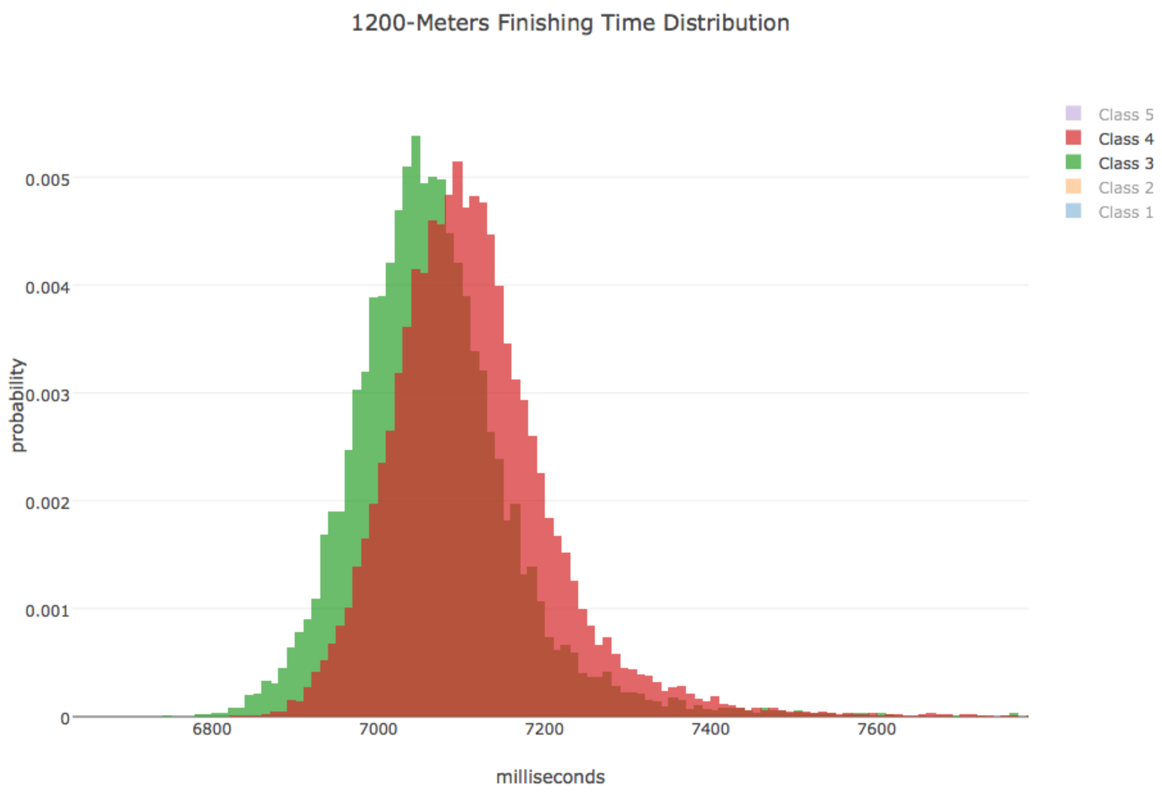


Figure 3

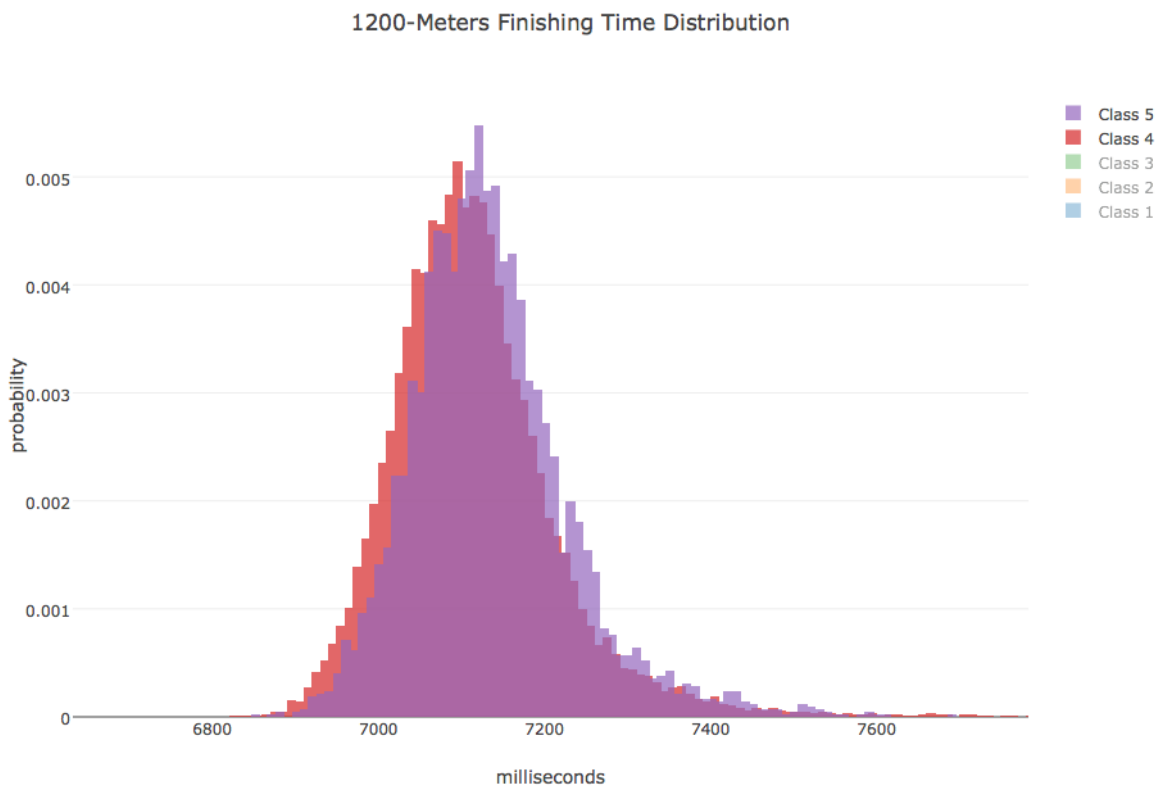


Figure 4

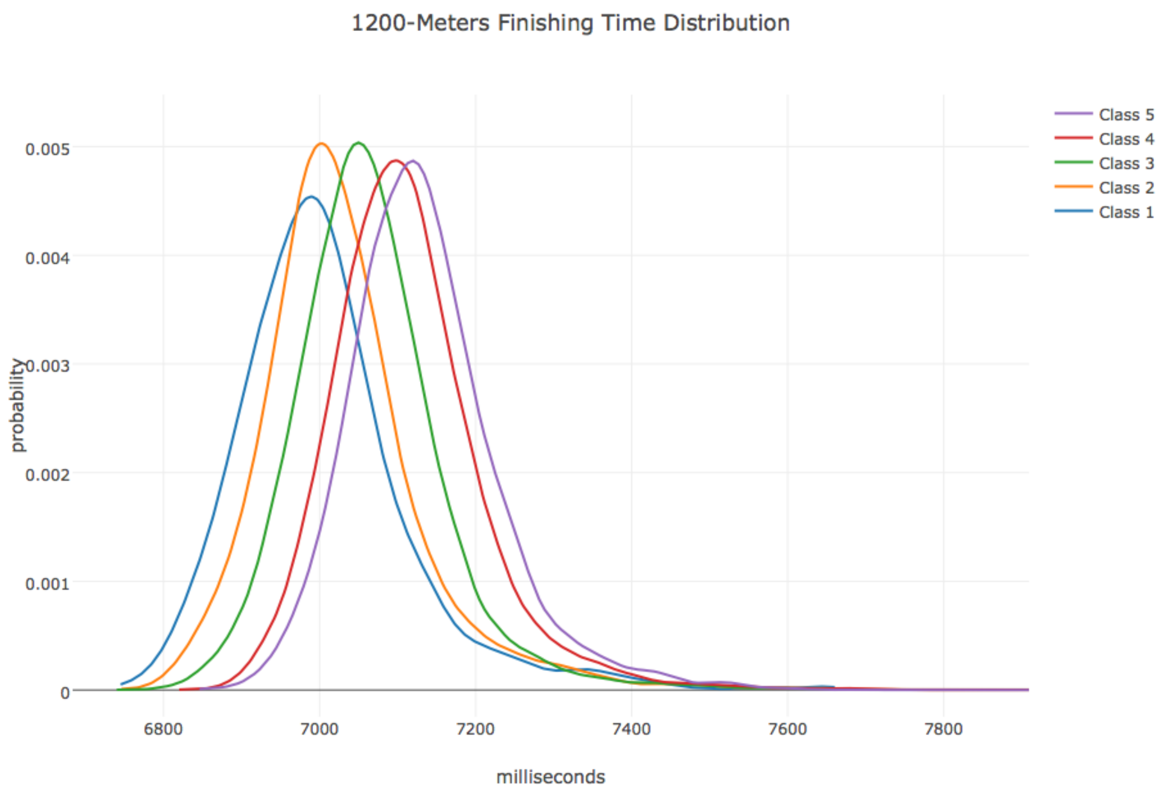


Figure 5

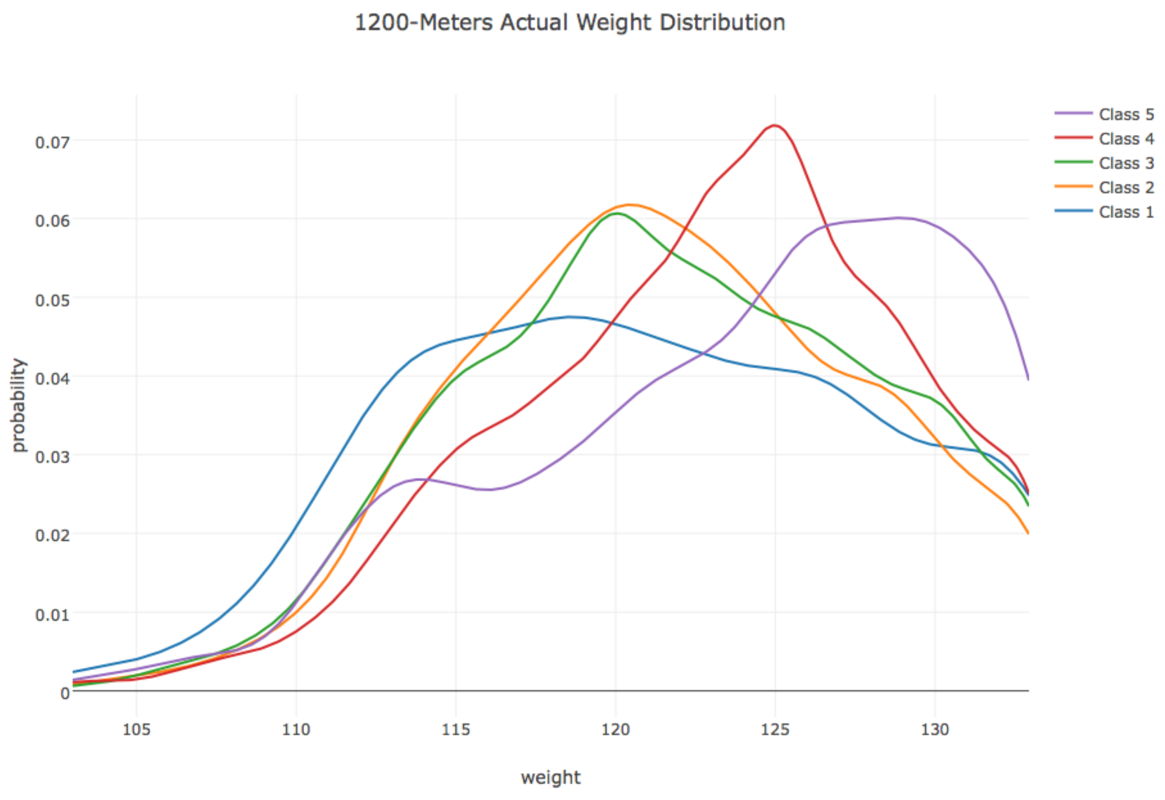


Figure 6

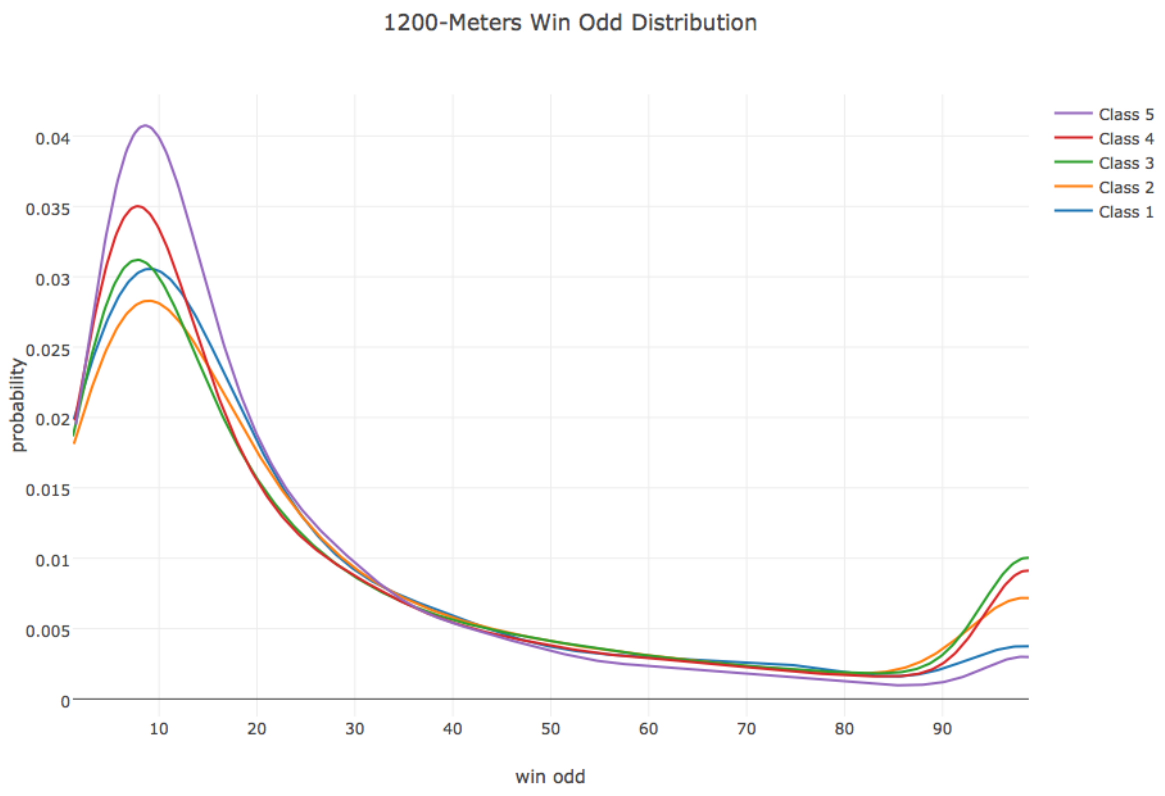


Figure 7

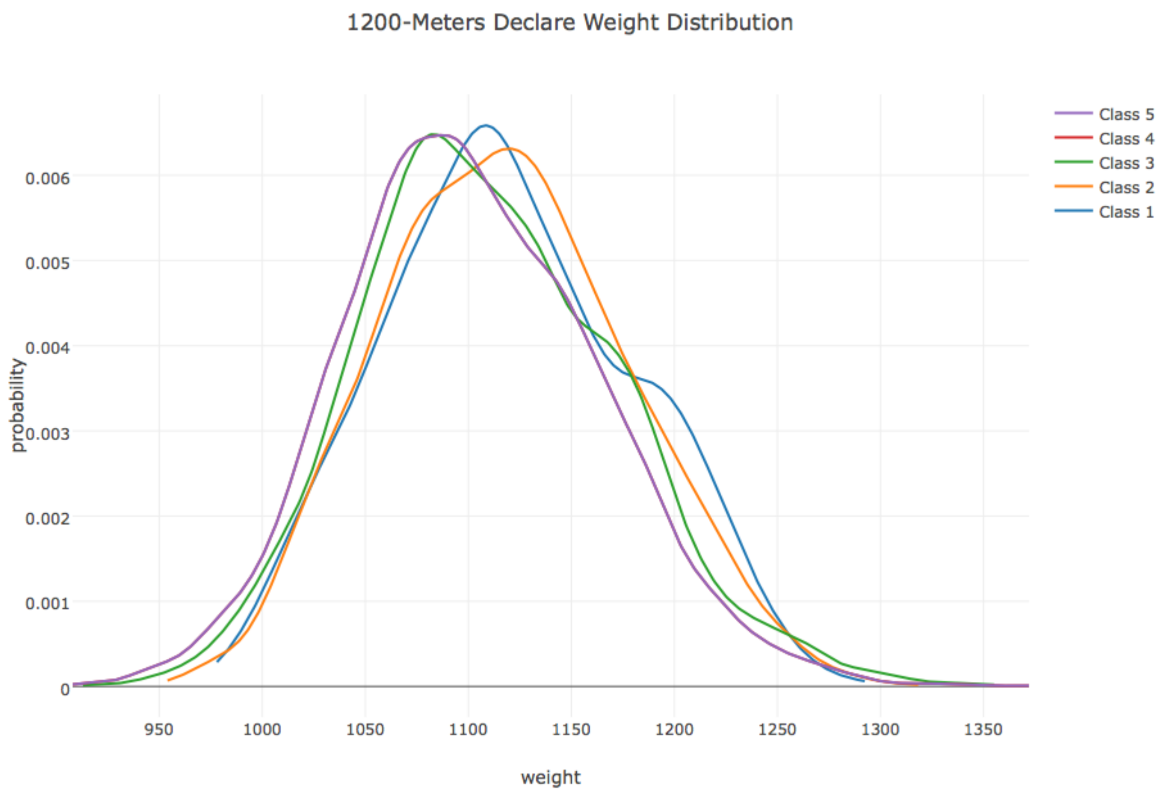


Figure 8

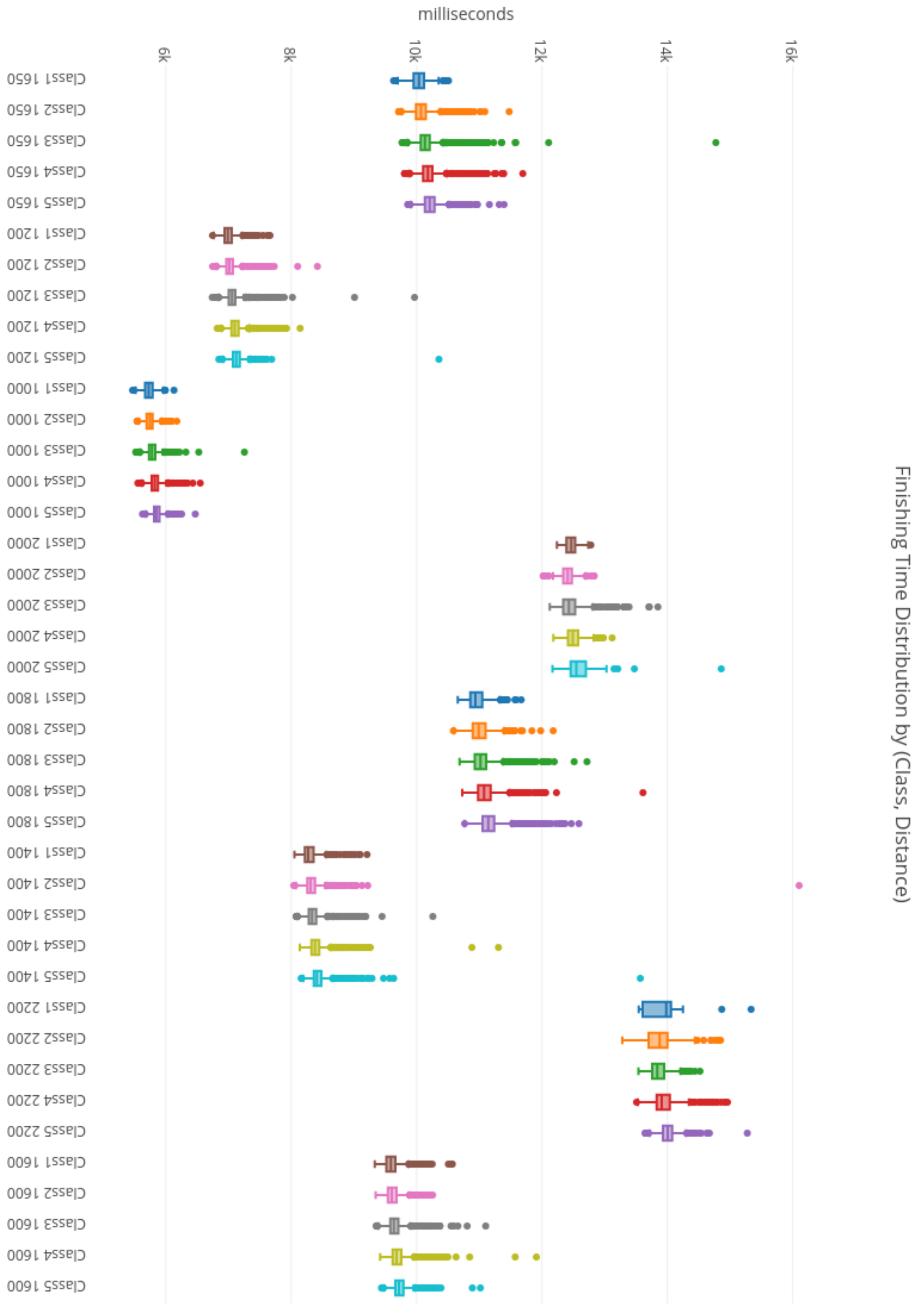

Finishing Time Distribution by (Class, Distance)
milliseconds
6k
8k
10k
12k
14k
16k
Class1 1650
Class2 1650
Class3 1650
Class4 1650
Class5 1650
Class1 1200
Class2 1200
Class3 1200
Class4 1200
Class5 1200
Class1 1000
Class2 1000
Class3 1000
Class4 1000
Class5 1000
Class1 2000
Class2 2000
Class3 2000
Class4 2000
Class5 2000
Class1 1800
Class2 1800
Class3 1800
Class4 1800
Class5 1800
Class1 1400
Class2 1400
Class3 1400
Class4 1400
Class5 1400
Class1 2200
Class2 2200
Class3 2200
Class4 2200
Class5 2200
Class1 1600
Class2 1600
Class3 1600
Class4 1600
Class5 1600

Figure 9

Implementation

Examples using classes and distance to describe situations has been introduced. However, a situation in a real scenario is more complex. It is because there are different horse racing tracks, which are also called course, and there are two locations for hosting horse racing in Hong Kong. Let's define a few mathematical notations.

Let x be the training dataset.

$$x = [x^{(1)} \quad x^{(2)} \quad \cdots \quad x^{(m)}]$$

where m is the total number of training data points.

Let $x^{(i)}$ be the i data points in x.

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$$

Let L be the set of locations, which included Sha Tin and Happy Valley, T be the set of courses, C be the set of classes, which include class 1 to 5 and other special classes and D be the set of distances. As $|L| = 2$, $|T| = 11$, $|C| = 11$ and $|D| = 9$, there are 2178 situations in total.

$$|L| \cdot |T| \cdot |C| \cdot |D| = 2178$$

Let $l \in L, t \in T, c \in C, d \in D$, a situation can now be described using l, t, c and d, where l, t, c, d are known information within a row record. Let $\hat{l}$ be the index of a row vector that is representing the location, $\hat{t}$ be the index of a row vector that is representing the course, $\hat{c}$ be the index of a row vector that is representing the class, $\hat{d}$ be the index of a row vector that is representing the distance. For any numerical feature i in the j row record, namely $x^{j}_{i_{l,t,c,d}}$.

$$x_{i_{l,t,c,d}}^{j} = \begin{cases} x_{\hat{l}}^{j} = l \\ x_{\hat{t}}^{j} = t \\ x_{\hat{c}}^{j} = c \\ x_{\hat{d}}^{j} = d \end{cases}$$

We standardize $x_{i_{l,t,c,d}}^{j}$by first calculating $\mu_{i_{l,t,c,d}}$.

$$\mu_{i_{l,t,c,d}} = \frac{1}{\left|S_{l,t,c,d}\right|} \sum_{k \in S_{l,t,c,d}} k_i$$

Then we calculate $\sigma_{i_{l,t,c,d}}$ by subtitling $\mu_{i_{l,t,c,d}}$.

$$\sigma_{i_{l,t,c,d}} = \sqrt{\left(\frac{1}{\left|S_{l,t,c,d}\right|} \sum_{k \in S_{l,t,c,d}} k_i^2\right) - \mu_{i_{l,t,c,d}}^2}$$

where $S_{l,t,c,d}$ is the set of row records that are satisficing the condition of l, t, c and d.

$$S_{l,t,c,d} = \left\{x^{(j)} \middle| x^{(j)} \in x,\ x_{\hat{l}}^{j} = l,\ x_{\hat{t}}^{j} = t\ ,\ x_{\hat{c}}^{j} = c,\ x_{\hat{d}}^{j} = d\right\}$$

After that we compute the standardized value $z_{i_{l,t,c,d}}^{j}$by subtracting $\mu_{i_{l,t,c,d}}$from $x_{i_{l,t,c,d}}^{j}$ and then divided by $\sigma_{i_{l,t,c,d}}$.

$$z_{i_{l,t,c,d}}^{j} = \frac{x_{i_{l,t,c,d}}^{j} - \mu_{i_{l,t,c,d}}}{\sigma_{i_{l,t,c,d}}}$$

Data Extraction

Past K records

The general belief that the more the data related to the past we can capture, the higher the chance we can predict the future correctly. In the field of horse racing, a horse will participate only around 15 – 20 race in its entire life (Figure 1.6). Therefore, the more the past records retained in a row vector, the fewer the data points for training.

Define h be the index of a row vector that is representing the horse's identity, r be the index of a row vector that is representing the race's identity. Define $\hat{h}$ be the horse's identity of a row vector, $\tilde{r}$ be the race's identity of a row vector.

$$\hat{h} = x_h^j$$
$$\hat{r} = x_{\tilde{r}}^j$$

Define $S_{\hat{h}}$ be the set of row vectors that are related to the horse's identity $\hat{h}$.

$$S_{\hat{h}} = \{x^j | x^j \in x,\ x_h^j = \hat{h}\}$$

Define $S_{\hat{r}}$ be the set of data points that are happened before the race with identity $\hat{r}$.

$$S_{\hat{r}} = \{x^j | x^j \in x,\ x_{\tilde{r}}^j < \hat{r}\}$$

Define $S_{\hat{p}}$ be the set of data points that are the past record for the horse with identity $\hat{h}$ with the race with identity $\tilde{r}$ as the origin.

$$S_{\hat{p}} = S_{\hat{h}} \cap S_{\hat{r}}$$

If $|S_{\hat{p}}| < k$, we drop the data points x^j. If $|S_{\hat{p}}| \geq k$, we append past k data points to x^j. Define p be the vector of past k recent data points sorted by race identities. Let $s_{\hat{p}}^{(n)}$ be the most recent data point and $s_{\hat{p}}^{(1)}$ be the last data point in $S_{\hat{p}}$.

$$p = \begin{bmatrix} s_{\hat{p}}^{(n)} & s_{\hat{p}}^{(n-1)} & \cdots & s_{\hat{p}}^{(n-k)} \end{bmatrix}$$

Then we define k_i as the new form of $x_i^{(j)}$.

$$k_i = \begin{bmatrix} p_i^{(1)} & \cdots & p_i^{(k)} & x_i^{(j)} \end{bmatrix}$$

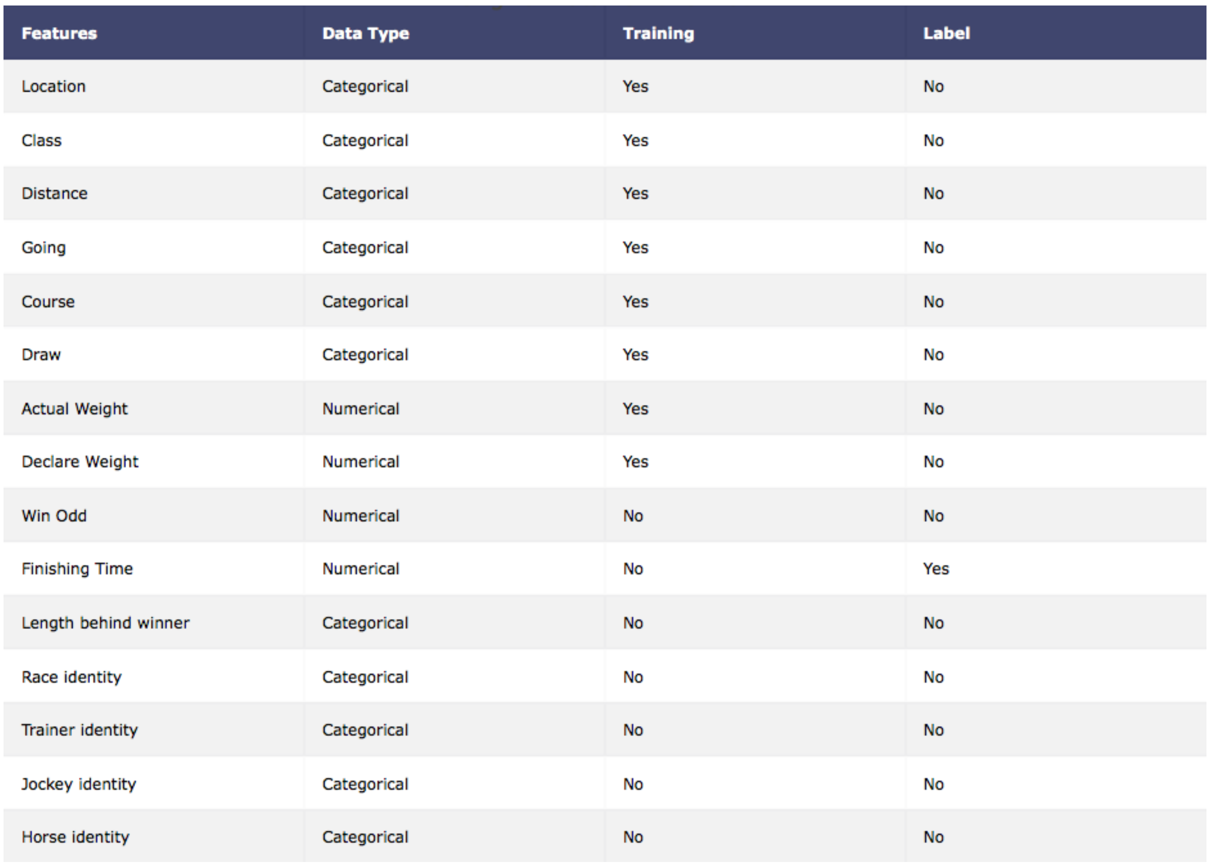
Set the new form of $x^{(j)}$ by concatenating k_is.

$$x^{(j)} \coloneqq k_1 \oplus k_2 \oplus \cdots \oplus k_n$$

where n is the number of features.

The original data point has 15 features (Table 1). The column of finishing time will be used as the training labels.

Table 1



Features	Data Type	Training	Label
Location	Categorical	Yes	No
Class	Categorical	Yes	No
Distance	Categorical	Yes	No
Going	Categorical	Yes	No
Course	Categorical	Yes	No
Draw	Categorical	Yes	No
Actual Weight	Numerical	Yes	No
Declare Weight	Numerical	Yes	No
Win Odd	Numerical	No	No
Finishing Time	Numerical	No	Yes
Length behind winner	Categorical	No	No
Race identity	Categorical	No	No
Trainer identity	Categorical	No	No
Jockey identity	Categorical	No	No
Horse identity	Categorical	No	No

After extracting k-past records, the k-past record temple (Table 2) will be added to the original data point k times. Therefore, the length of original data point will be increased from 15 to 15 times (k+1).

Table 2

Features	Data Type	Training	Label
Location (k)	Categorical	No	No
Class (k)	Categorical	No	No
Distance (k)	Categorical	No	No
Going (k)	Categorical	No	No
Course (k)	Categorical	No	No
Draw (k)	Categorical	No	No
Actual Weight (k)	Numerical	Yes	No
Declare Weight (k)	Numerical	Yes	No
Win Odd (k)	Numerical	No	No
Finishing Time (k)	Numerical	Yes	No
Length behind winner (k)	Categorical	No	No
Race identity (k)	Categorical	No	No
Trainer identity (k)	Categorical	No	No
Jockey identity (k)	Categorical	No	No
Horse identity (k)	Categorical	No	No

Amount of records VS K-value

As a horse will only participate 15 – 20 race in average, the more past records we would like to capture, the fewer complete row records for us to use. A complete row record means a row contains the complete information of the past-k records with respect to the horse's identity. Therefore, the higher the k-value, the fewer the training data (Figure 10).

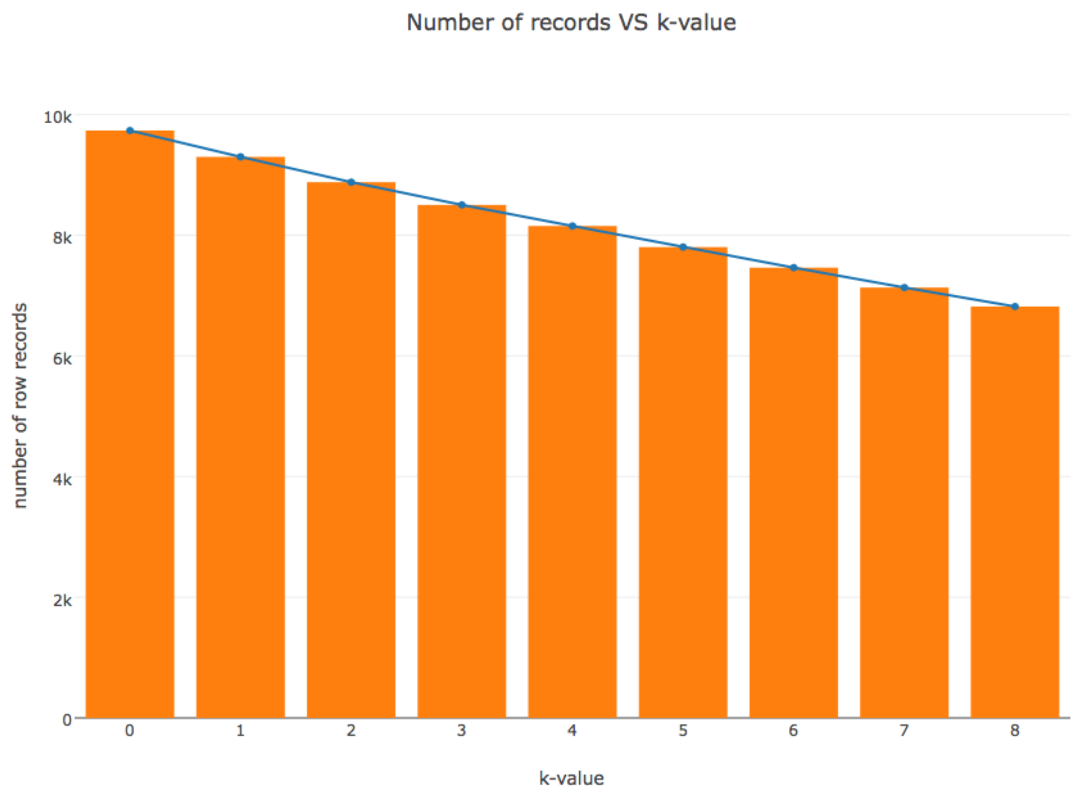


Figure 10

Model

Linear Regression

Last semester, we model the problem as a classification problem, which is also called logistic regression problem, which was predicting the probability of a horse to win a race. In this semester, we would like to re-define the problem as a regression problem, which is predicting the finishing time of a horse in a race.

Let $h(x)$ be the linear regression model that used in this semester.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

Let θ be the vector of parameters of the learning model. Let x be the vector of data points, where $x_0 = 1$ and n is the number of features.

$$\theta = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n]$$

$$x = [x_0 \quad x_1 \quad \cdots \quad x_n]$$

For simplicity, $\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$ can be written as a product of vectors.

$$\theta^{\mathrm{T}} x = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} [x_0 \quad x_1 \quad \cdots \quad x_n] = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

And $h(x)$ can be written as following.

$$h_\theta(x) = \theta^{\mathrm{T}} x$$

The different of the learning model between this semester, which is $h(x)$, and the last semester, which is $\hat{h}(x)$, is about the application of the $sigmoid$ function. The $sigmoid$

function (Figure 11) accepts value range from $-\infty$ to $+\infty$ and the output boundaries of a $sigmoid$ function is $(0, 1)$. Therefore, it turns a regression function to a logistic function.

$$\hat{h}_\theta(x) = sigmoid(\theta^\mathrm{T} x) = \frac{1}{1 + e^{-\theta^\mathrm{T} x}}$$

The model we have been using this semester is simpler than the last one as we dropped the $sigmoid$ function.

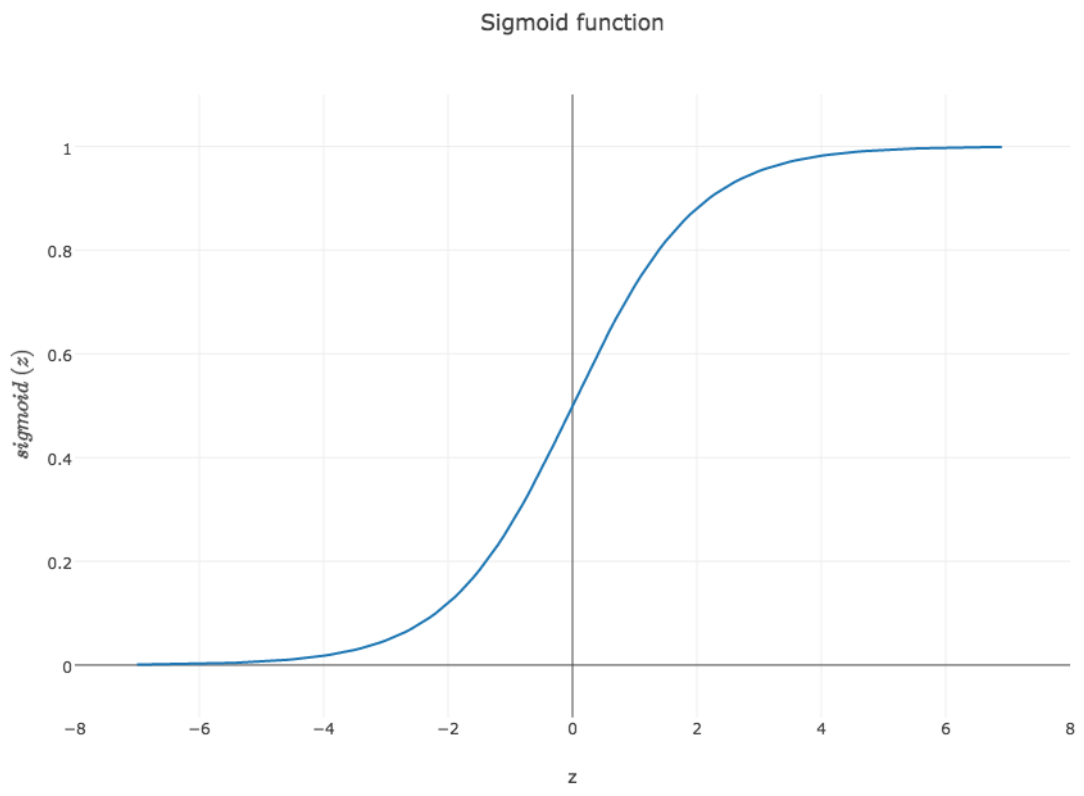


Figure 11

Now we have our learning model $h_\theta(x) = \theta^{\mathrm{T}}x$ and y which is a vector of training labels, which are finishing time. Define $J(\theta)$ be the loss function of the learning model, which is also called "mean square error function", where m is the total number of training data points.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Our goal is find $\hat{\theta}$ such that it would minimize the error function $J(\theta)$.

$$\hat{\theta} = \min_{\theta} J(\theta)$$

Traditional, we can use gradient descent algorithm to find $\hat{\theta}$,

$$\theta_i \coloneqq \theta_i - \eta \nabla J(\theta)$$

where η is the learning factor of the algorithm. Repeat the above algorithm for $i = 1, 2, \cdots, n$ until convergence.

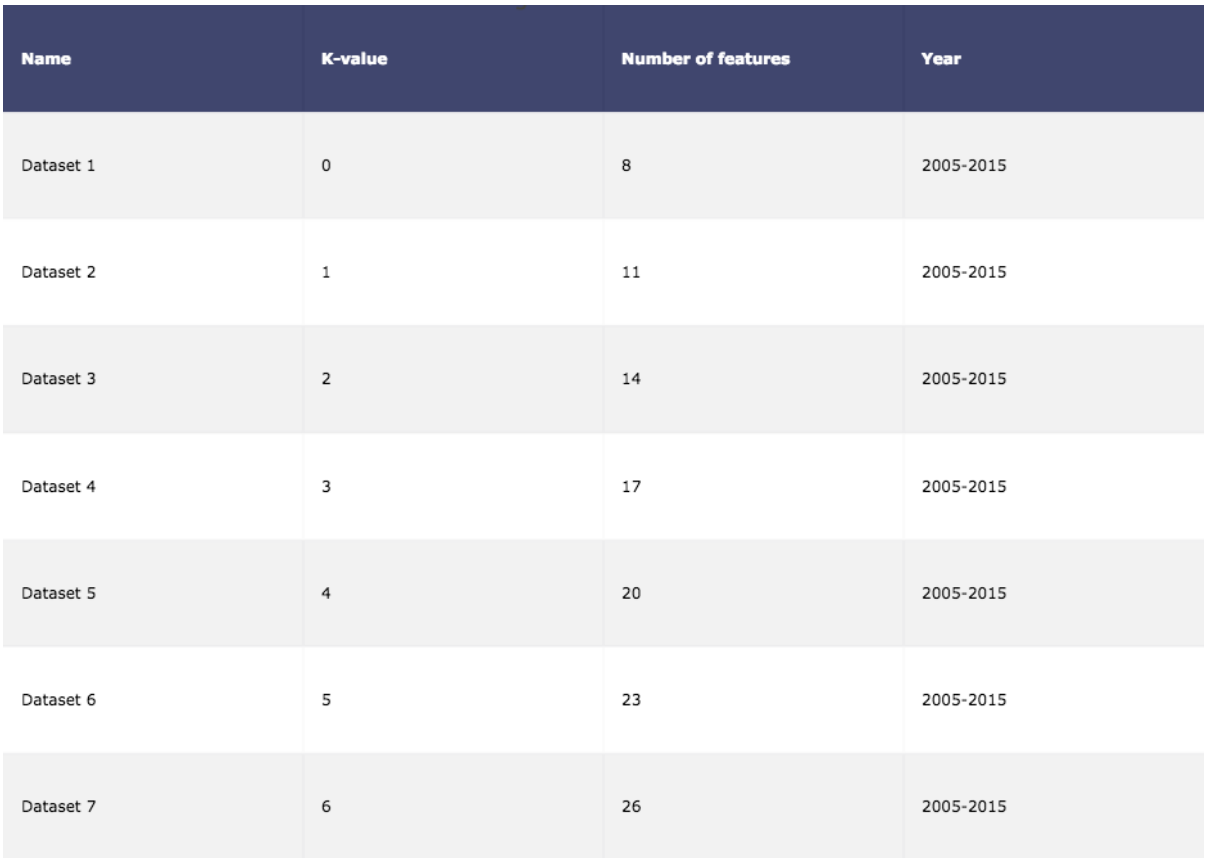
Tensorflow linear estimator

However, thanks to Tensorflow, we don’t have to worry about the implementation of the gradient descent algorithm as Tensorflow already did it. We can even apply fancy optimization algorithms that were built-in in Tensorflow library, such as AdaGrad optimization algorithm, Adam optimization algorithm and Follow the Regularized Leader algorithm. As the default optimization algorithm of the linear regression model of Tensorflow is Follow the Regularized Leader (FTRL) algorithm, we were using FTRL algorithm instead of gradient descent to do the optimization.

Training

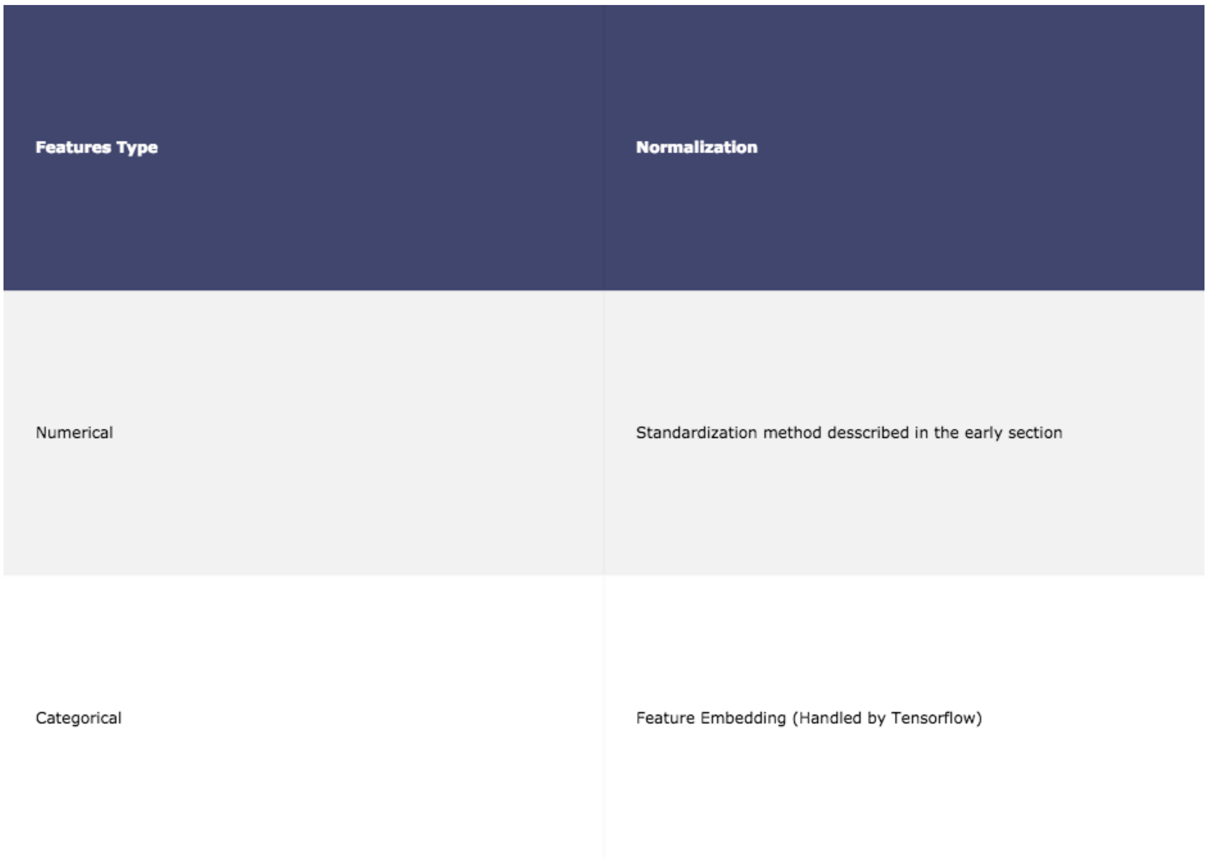
We were using 2005 – 2015 horse racing data for the training. We had prepared 7 set of data, where each set of data are containing K past records, such that K is from 0 to 6 respectively (Table 3). Numerical features were normalized using the method we have introduced and categorical features were handled by Tensorflow automatically (Table 4). After that we had trained a model for each dataset respectively with 1000 epochs and Follow the Regularized Leader optimizer, and with the learning rate $\frac{1}{\min(0.2,\sqrt{n})}$, where n is the number of features (Table 5).

Table 3



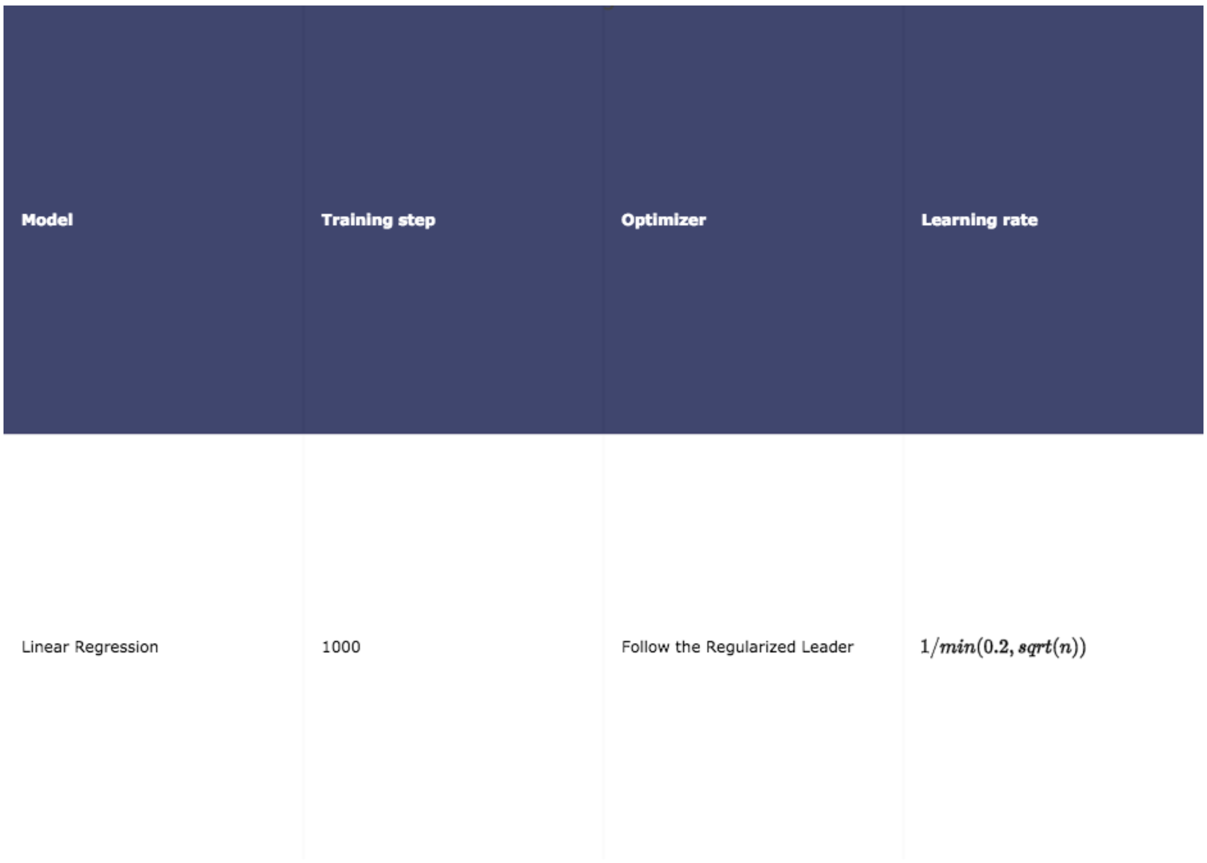
Name	K-value	Number of features	Year
Dataset 1	0	8	2005-2015
Dataset 2	1	11	2005-2015
Dataset 3	2	14	2005-2015
Dataset 4	3	17	2005-2015
Dataset 5	4	20	2005-2015
Dataset 6	5	23	2005-2015
Dataset 7	6	26	2005-2015

Table 4



Features Type	Normalization
Numerical	Standardization method desscribed in the early section
Categorical	Feature Embedding (Handled by Tensorflow)

Table 5



Model	Training step	Optimizer	Learning rate
Linear Regression	1000	Follow the Regularized Leader	$1/min(0.2, sqrt(n))$

Result

As a result, the higher the k-value the lower the loss value after 1000 epochs (Figure 12). However, the higher the k-value, the longer the time required for the training (Figure 13). It is a tradeoff between training time and loss value. In figure 13, the x-axis is the relative time, we can see that the lower the k-value the faster the time for the model to finish 1000 epochs training.

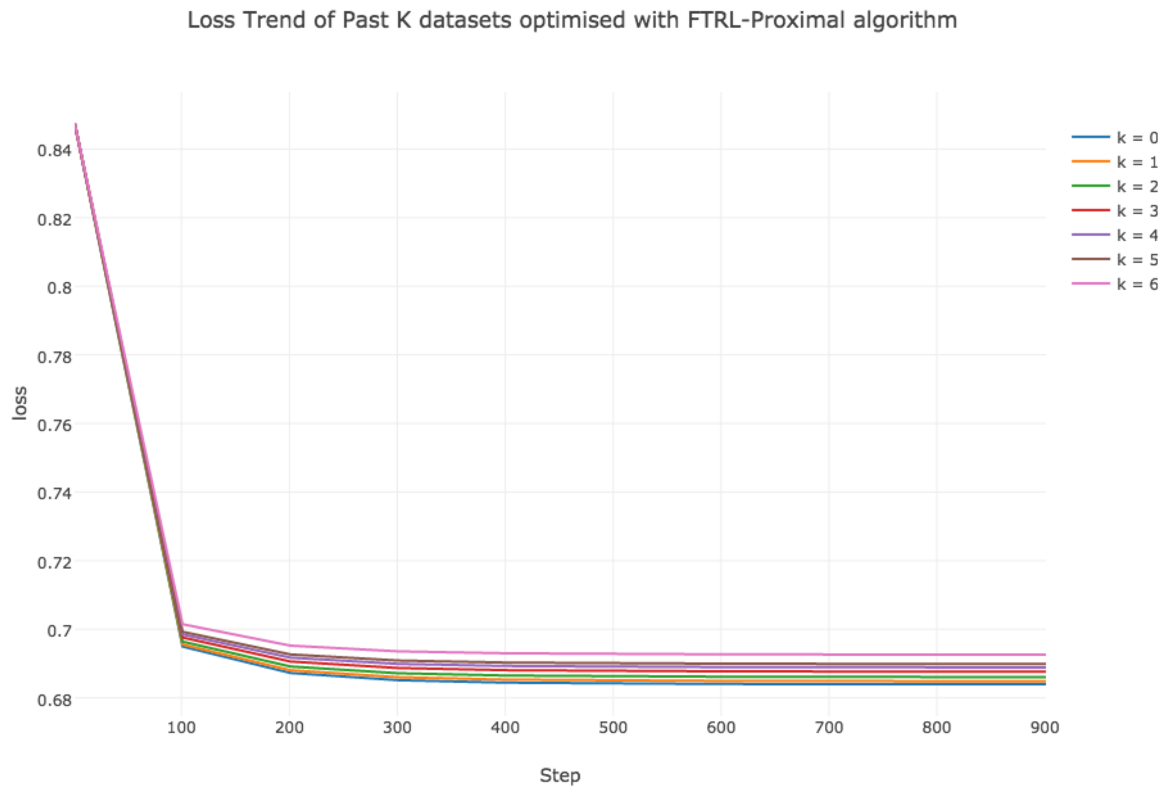


Figure 12

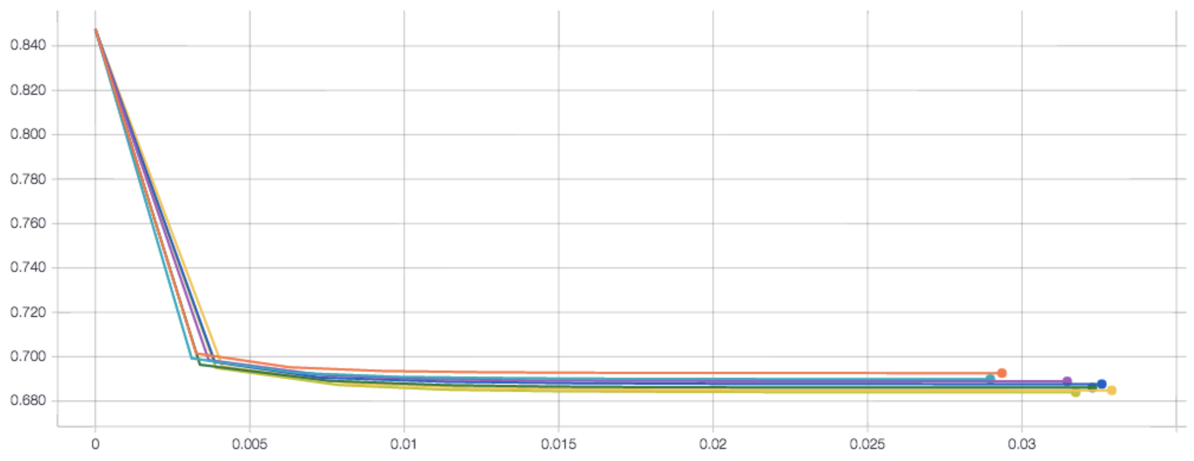
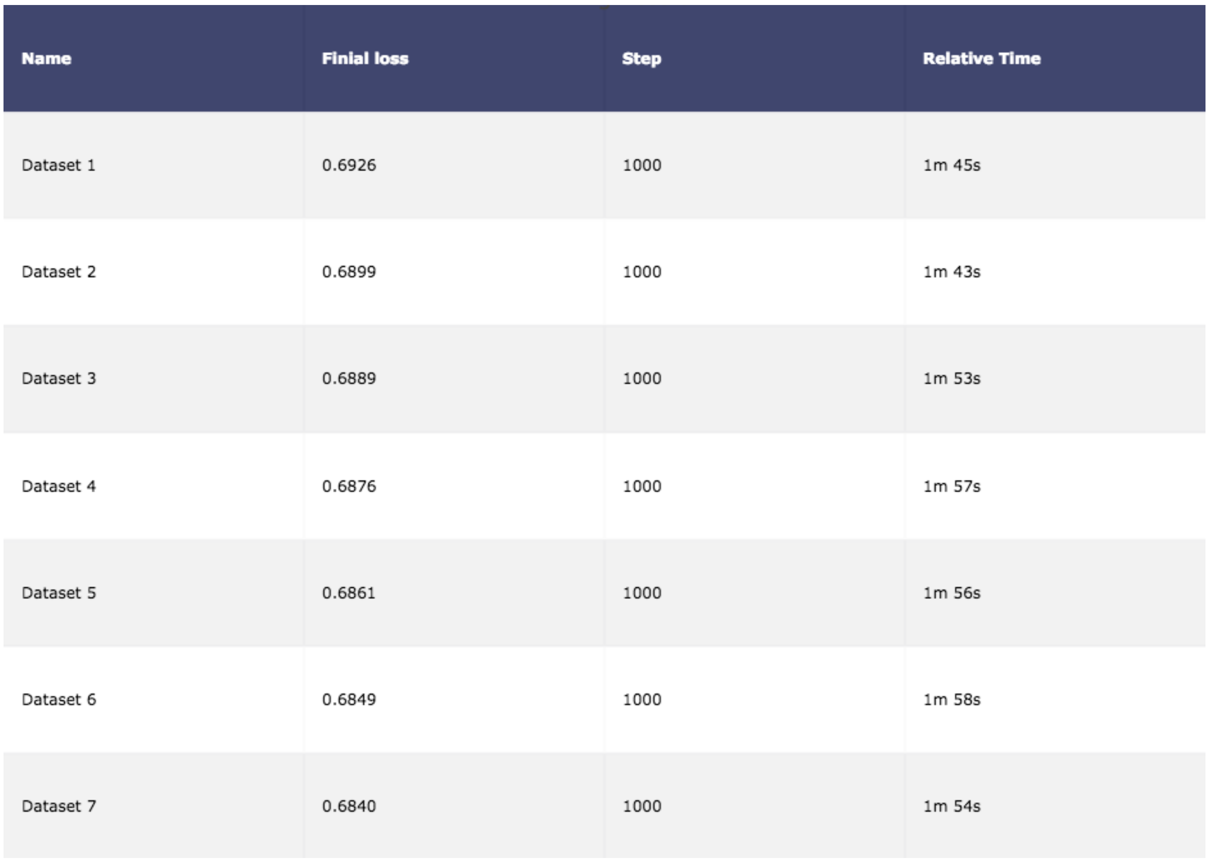


Figure 13

In general, we can agree that the higher the k-value, the best the result we will get. However, the amount of data is decreasing when the k-value is increasing (Figure 10). Maybe the k-value raised to a certain point, the loss value may stop decreasing and start to increase instead. For now, the critical point is definitely less than 6 as the trend of the loss was decreasing, though it took longer time to train datasets with high k-value, the difference of training time among datasets were small (Table 6).

Table 6



Name	Finial loss	Step	Relative Time
Dataset 1	0.6926	1000	1m 45s
Dataset 2	0.6899	1000	1m 43s
Dataset 3	0.6889	1000	1m 53s
Dataset 4	0.6876	1000	1m 57s
Dataset 5	0.6861	1000	1m 56s
Dataset 6	0.6849	1000	1m 58s
Dataset 7	0.6840	1000	1m 54s

We had 7 trained models and we evaluated each trained models using 2015 – 2016 dataset and 2016 – 2017 February respectively (Table 7). The result is as expected, the higher the k value, the lower the evaluation loss, though the different of the evaluation loss and the loss value of training is huge, 2015 – 2016 and 2016 – 2017 February datasets were agreeing on the same result (Figure 14).

Table 7

Name	Evaluation loss of 2015-16	Evaluation loss of 2016-17
Dataset 1	0.888349	0.885632
Dataset 2	0.877065	0.8662
Dataset 3	0.870756	0.856957
Dataset 4	0.868337	0.849918
Dataset 5	0.865304	0.844879
Dataset 6	0.862173	0.840881
Dataset 7	0.859776	0.839384

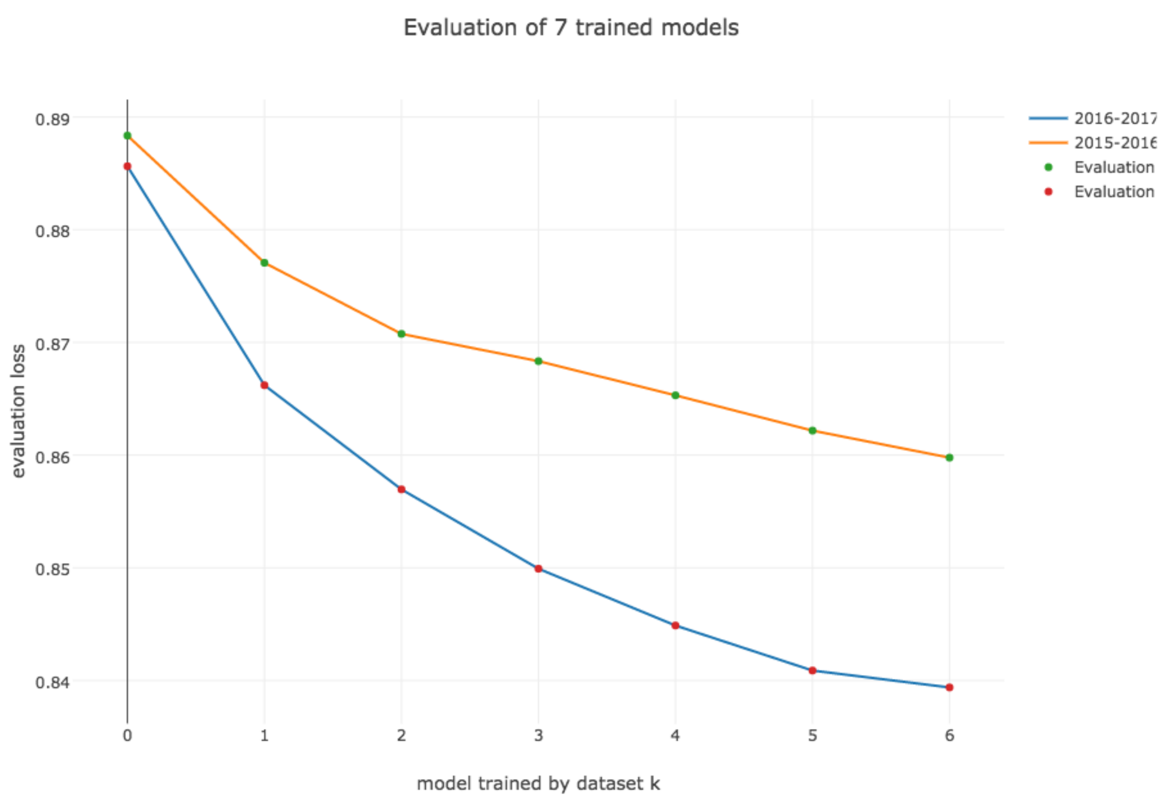


Figure 14

Evaluation

Evaluation by Race

Evaluate by race is different from the evaluation by value discussed in the previous chapter. In the previous chapter, the evaluation method is computing the mean square error of the predicted result and the actual result. The mean square error loss is only an indicator to show the performance of predicting the finishing time of a horse, but not the profitability of applying the model in a race. In this chapter, we are going to apply the model in real race, which means we are going to simulate different kinds of betting by making betting decisions based on the predicted results.

Assume we have unseen race records, which are testing dataset (2015 – 2016 and 2016 – 2017); As we have trained k+1 models where $k = 0, 1, \dots, 6$, we have to decide which model we are going to use, which means we have to decide the k value. After that we extract the testing dataset with past-k records by the procedure we have discussed in the early chapter. Now each row vector in the testing dataset with past-k records should have $b + kc$ features, where b is the number of features of the original vector and c is the number of features of the temple vector. In our case, b is 15 and c is 3.

$$\left|x^{(j)}\right| = 3k + 15$$

Before group the row vectors by race identities, we append the predicted results to each row vector correspondingly.

$$x^{(j)} = x^{(j)} \oplus h(x)^{(j)}$$

Denote R be the set of race such that R_i is a race with identity i.

$$R_i \in R$$

Then we have to group our testing data by the race's identity. Define $R_{\hat{r}}$ be the set of row vectors where each row vector in the set has the race's identity $\hat{r}$.

$$R_{\hat{r}} = \left\{ x^{(j)} \middle| \, x^{(j)} \in x, \; x_{\tilde{r}}^{(j)} = \hat{r} \right\}$$

where $\tilde{r}$ is the index of the race's identity in a row vector.

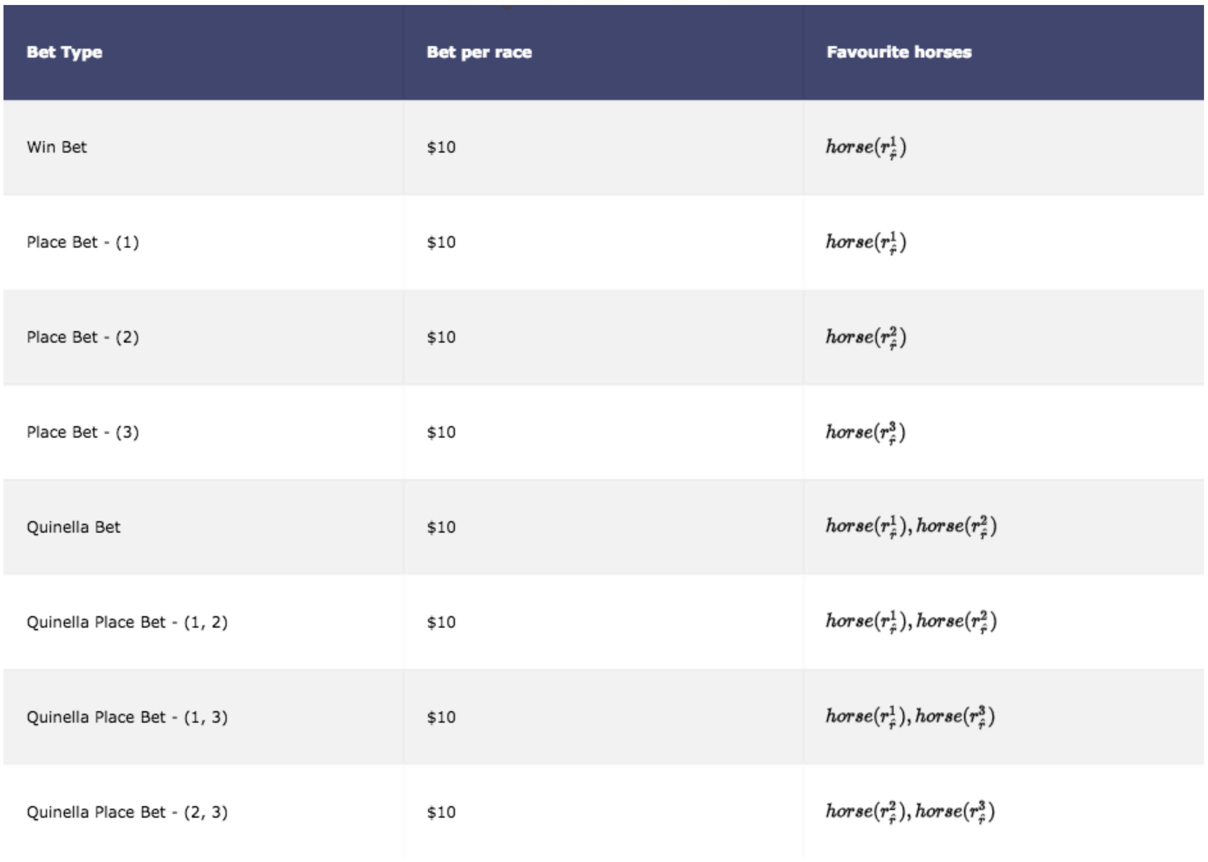
Since we may have dropped row vectors that don't have past-k records, we may have incomplete race information on our hands, therefore we have to filter the race that has missing row vectors, which means we retain race that have complete information.

$$R \coloneqq \{R_{\hat{r}} \in R | \, V_{\hat{r}} = |R_{\hat{r}}|\}$$

where $V_{\hat{r}}$ is the actual number of participant in the race with identity $\hat{r}$. What we are doing here is to retain the race that had the exact number of participant comparing with the actual one.

Denote $r_{\hat{r}}^{(i)}$ be the row vector corresponding to the i fastest horse in a race with identity $\hat{r}$. Then we can make the decision about which horse we are going to bet on based on the order of row vectors (Table 8). Define $horse(r_{\hat{r}}^{(i)})$ be a function of $r_{\hat{r}}^{(i)}$, such that it will output the horse's identity of the input row vector. We will bet \$10 on each race for each betting type respectively, and we will evaluate models by the net gain.

Table 8



Bet Type	Bet per race	Favourite horses
Win Bet	\$10	$horse(r_{\hat{r}}^{1})$
Place Bet - (1)	\$10	$horse(r_{\hat{r}}^{1})$
Place Bet - (2)	\$10	$horse(r_{\hat{r}}^{2})$
Place Bet - (3)	\$10	$horse(r_{\hat{r}}^{3})$
Quinella Bet	\$10	$horse(r_{\hat{r}}^{1}), horse(r_{\hat{r}}^{2})$
Quinella Place Bet - (1, 2)	\$10	$horse(r_{\hat{r}}^{1}), horse(r_{\hat{r}}^{2})$
Quinella Place Bet - (1, 3)	\$10	$horse(r_{\hat{r}}^{1}), horse(r_{\hat{r}}^{3})$
Quinella Place Bet - (2, 3)	\$10	$horse(r_{\hat{r}}^{2}), horse(r_{\hat{r}}^{3})$

Result

Normal Bet

Win Bet

The model performed really bad on 2015 – 2016 (Figure 15) and 2016 – 2017 (Figure 16). However, we can agree that the higher the k-value, the better the model will perform, though the order may not be absolute.

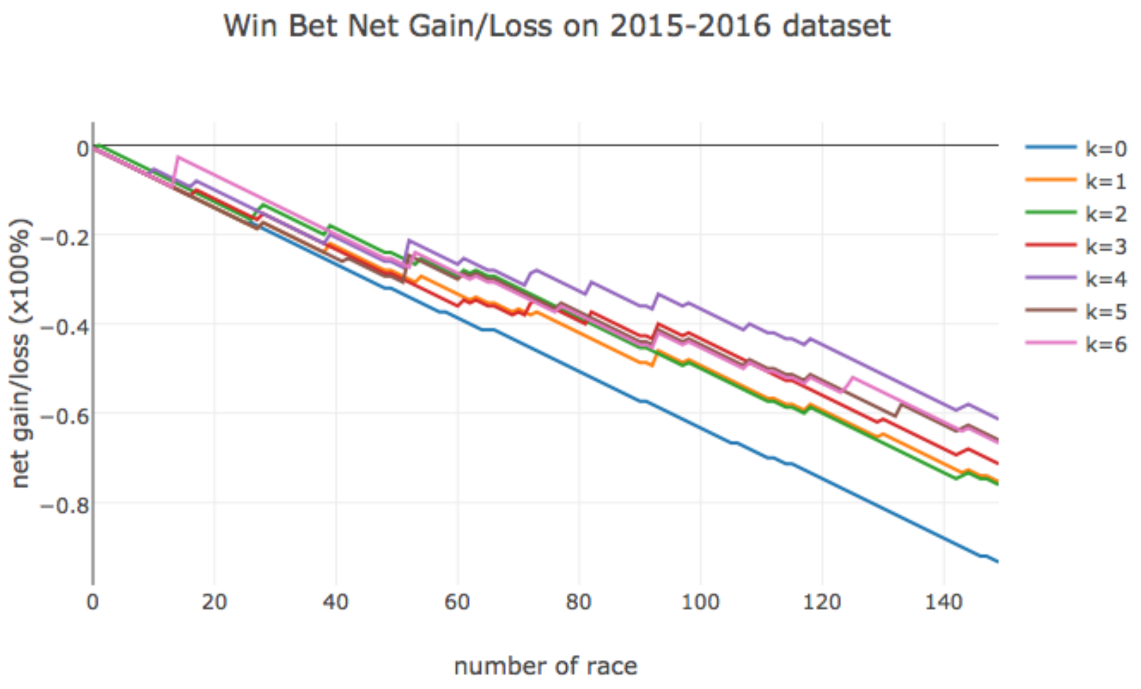


Figure 15 Simulate win bet on 2015-2016 race with complete information

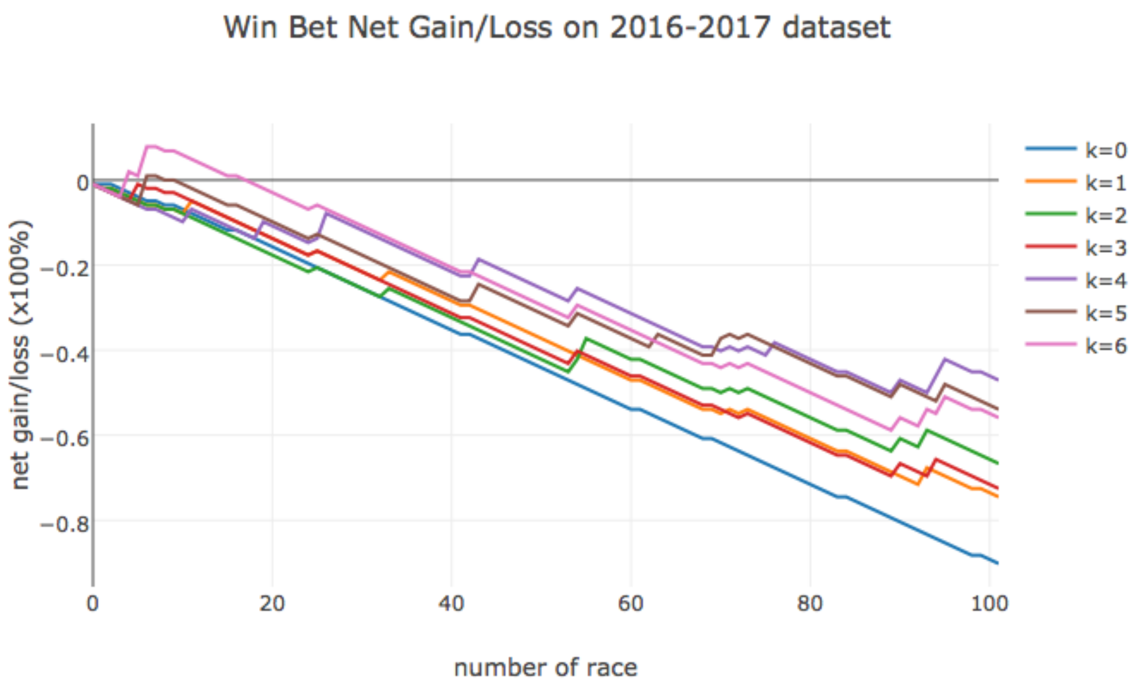


Figure 16 Simulate win bet on 2016-2017 race with complete information

Place Bet – (1)

The models perform not good in place bet – (1) either. However, again we can generally agree that the model performed better with a bigger k-value, although it is not absolutely true, it is generally true.

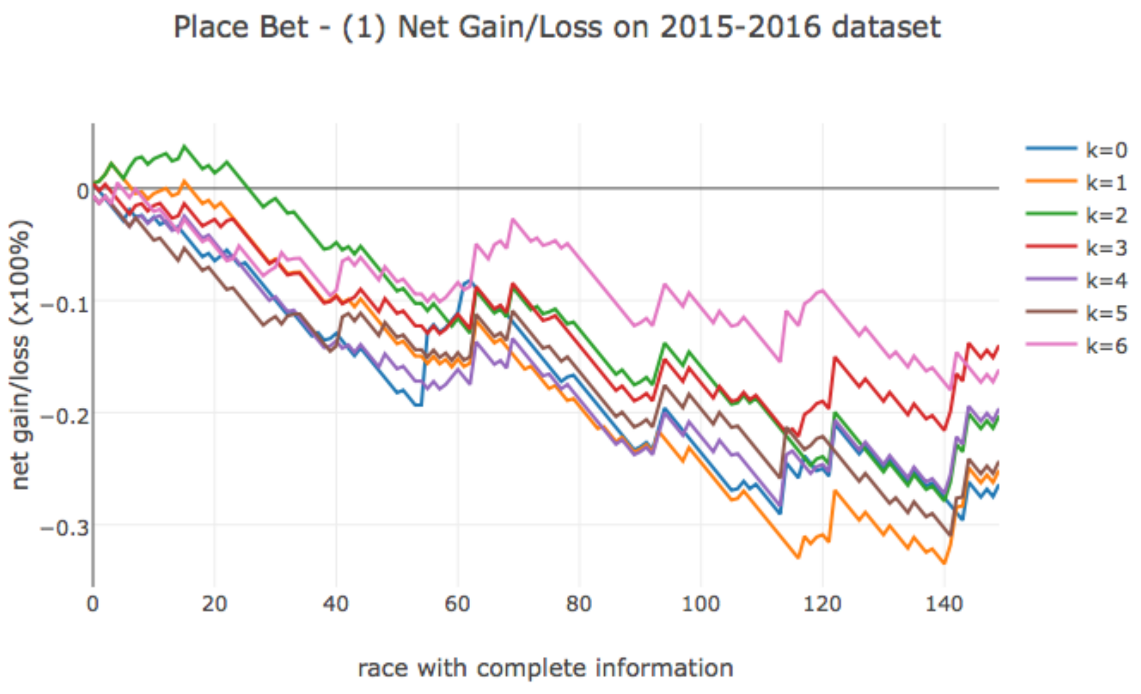


Figure 17 Simulate place bet – (1) on 2015-2016 race with complete information

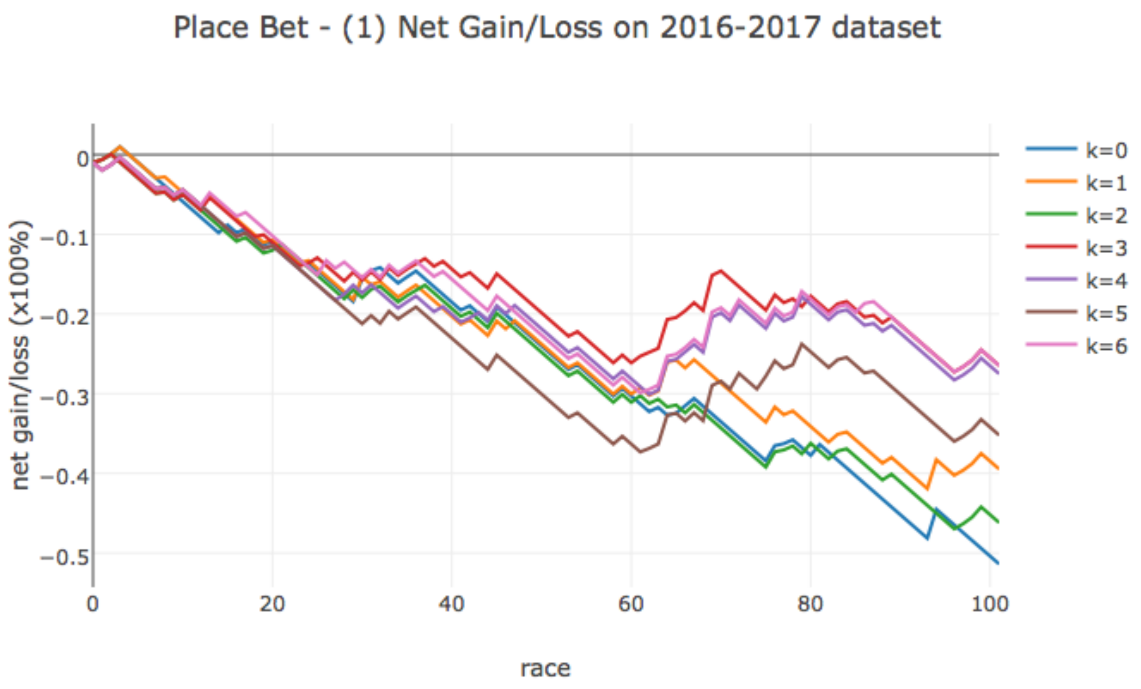


Figure 18 Simulate place bet – (1) on 2016-2017 race with complete information

Place Bet – (2)

The models perform better on Place Bet – (2) than Place Bet – (1) and Win Bet relatively. In 2015 – 2016, neither of the models yield a net gain (Figure 19). However, generating net gains are possible in 2016 – 2017 for the models with parameter k equals to 6 and 7 (Figure 20).

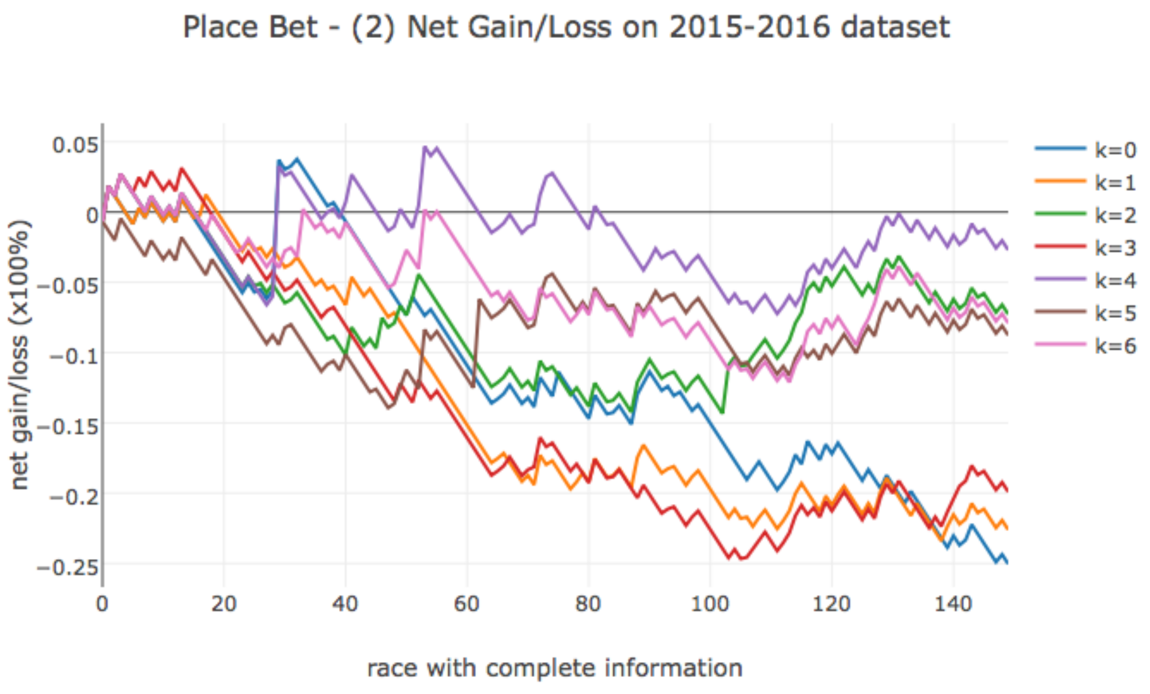


Figure 19 Simulate place bet – (2) on 2015-2016 race with complete information

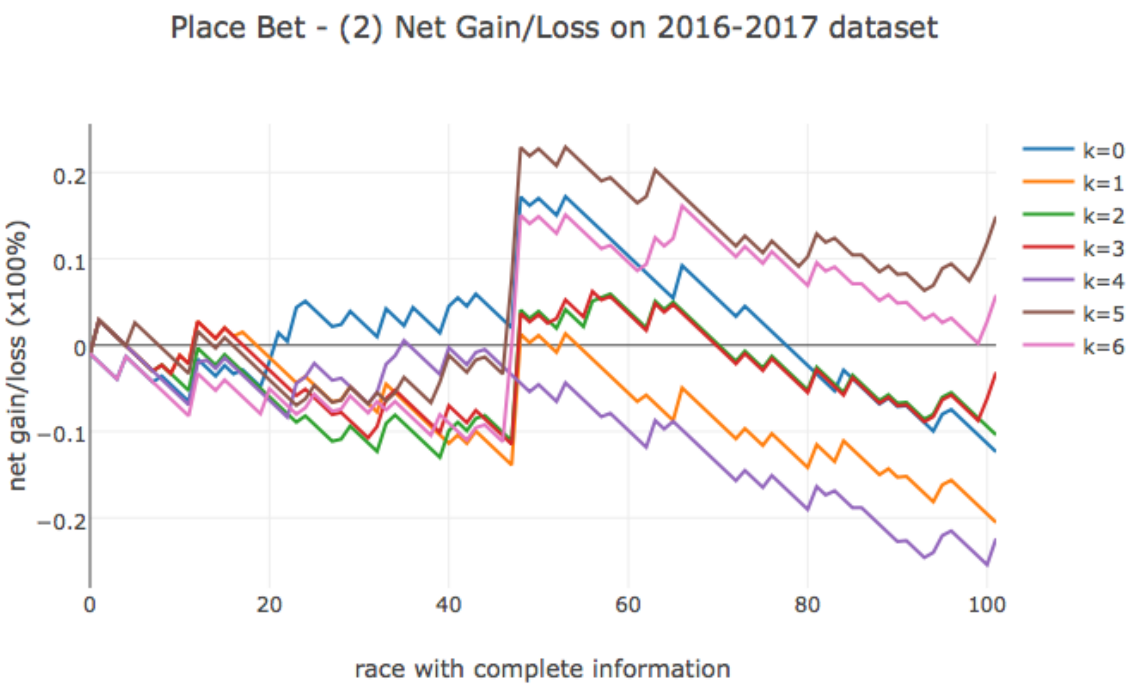


Figure 20 Simulate place bet – (2) on 2016-2017 race with complete information

Place Bet – (3)

The models perform better in Place Bet – (3) in general, when the result is comparing with Place Bet – (1) and Place Bet – (2). The models with parameter k equals to 0 and 1 could generate net gain in 2015 – 2016 (Figure 21). The models with parameter k equals to 0 and 1 again could generate net gain in 2016 – 2017 (Figure 22). However, the result is violating the agreement of "higher k-value, better value".

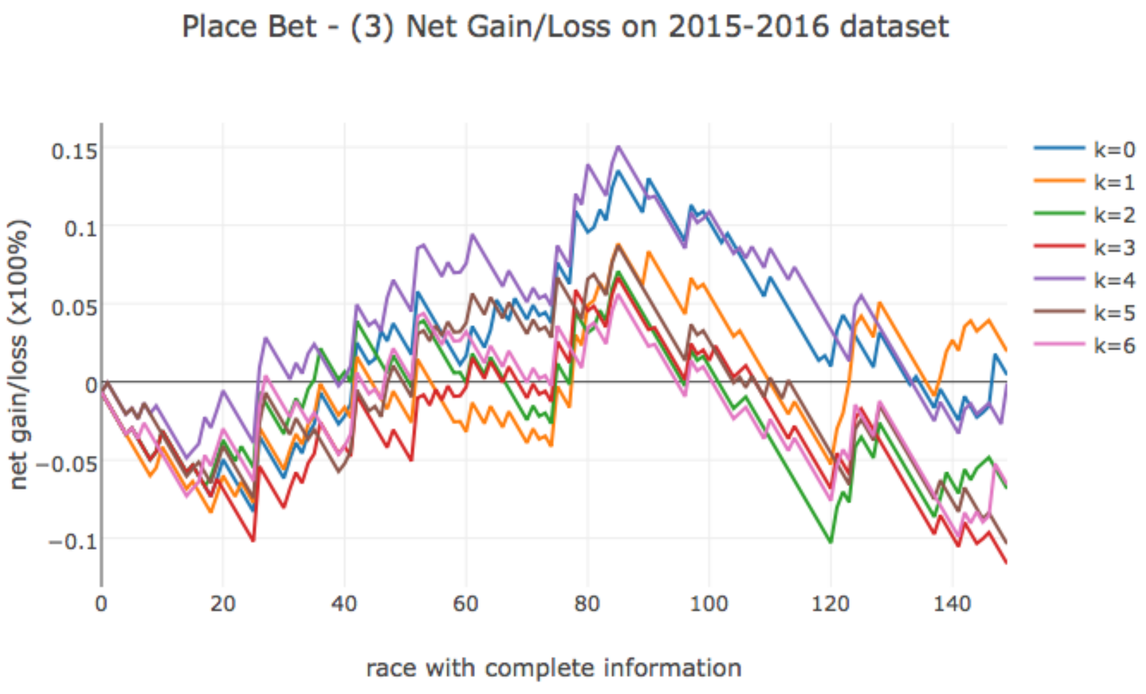


Figure 21

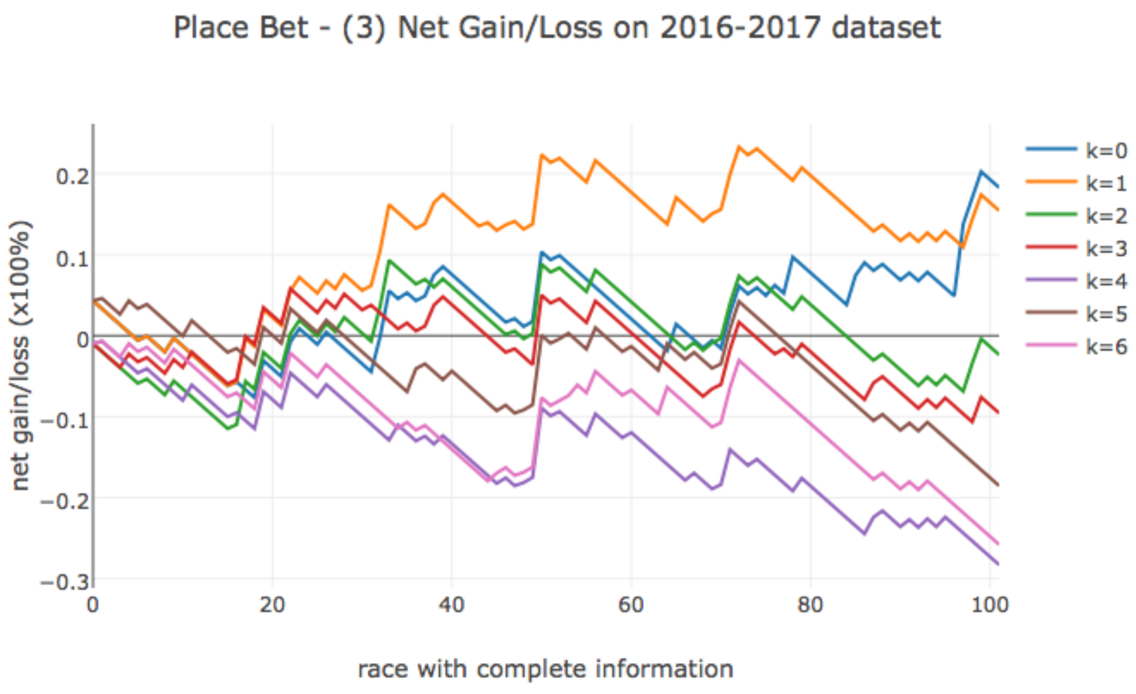

Place Bet - (3) Net Gain/Loss on 2016-2017 dataset
net gain/loss (x100%)
0.2
0.1
0
−0.1
−0.2
−0.3
0
20
40
60
80
100
race with complete information
k=0
k=1
k=2
k=3
k=4
k=5
k=6

Figure 22

Quinella Bet

The models perform really good in Quinella Bet. In 2015 – 2016 (Figure 23), the model with parameter k equals to 2, 3, 4, 5 and 6 could generate net gain, and if we sort them by the value of net gain, the order is preserved. In 2016 – 2017 (Figure 24), only the model with parameter k equals to 4 and 5 could have net gain.

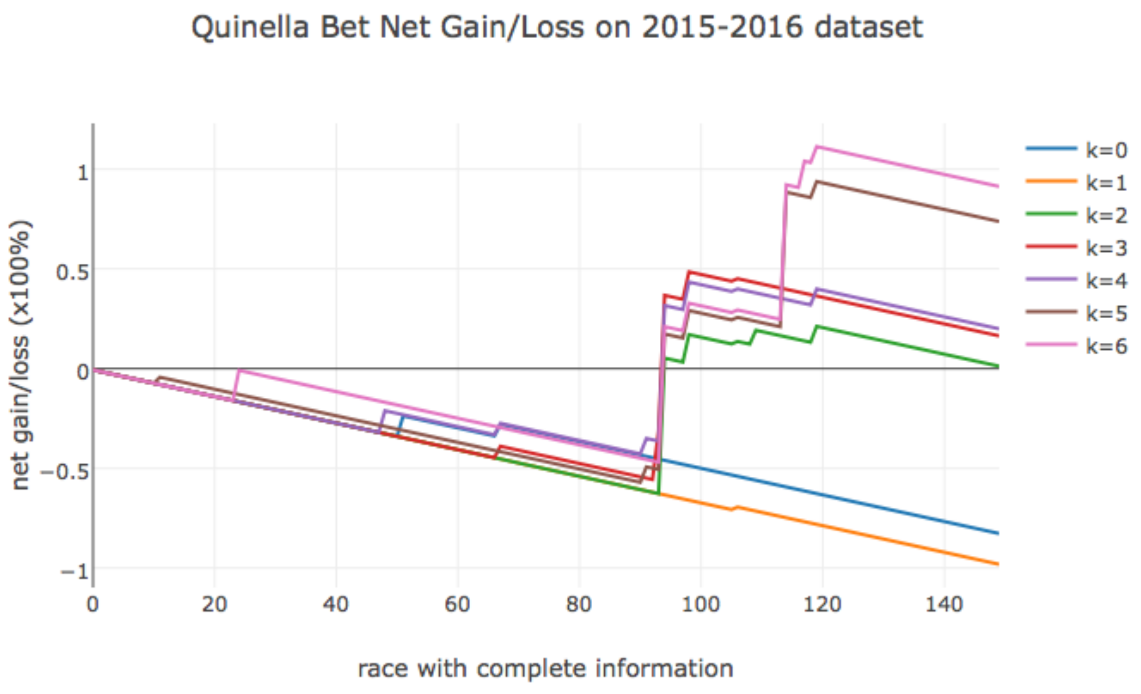


Figure 23

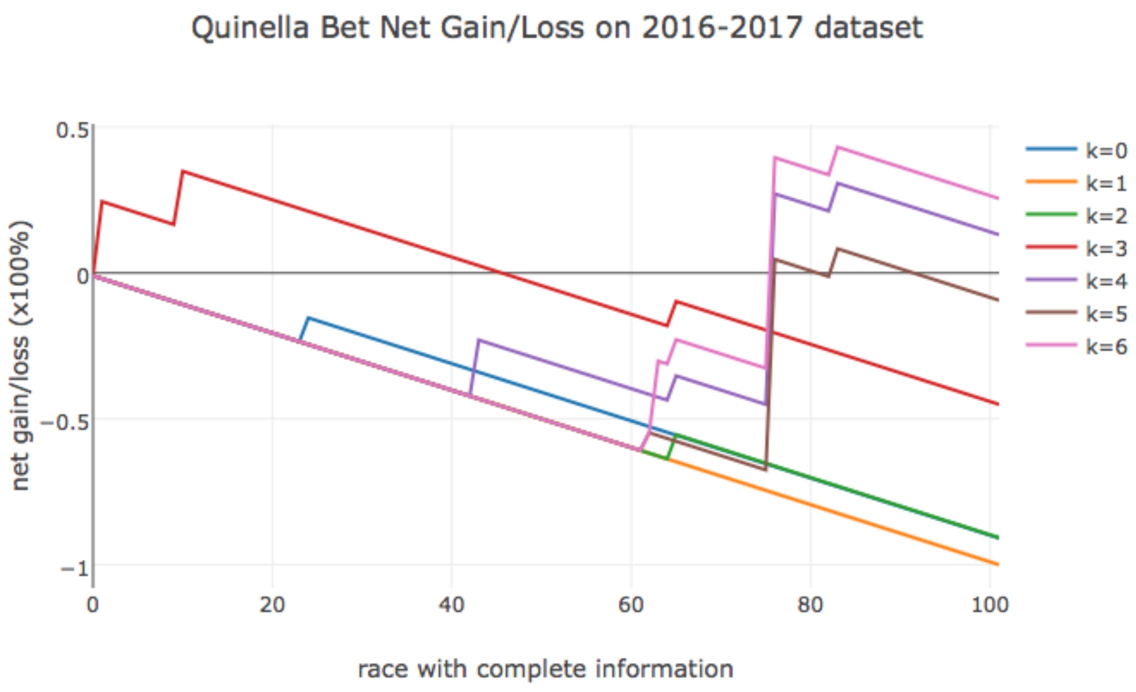

Quinella Bet Net Gain/Loss on 2016-2017 dataset
net gain/loss (x100%)
0.5
0
−0.5
−1
0
20
40
60
80
100
race with complete information
k=0
k=1
k=2
k=3
k=4
k=5
k=6

Figure 24

Quinella Place Bet – (1, 2)

The models perform bad on Quinella Place Bet – (1, 2) comparing with Quniella Bet. Only models with parameter k equals to 5 and 6 could have net gain in 2015 – 2016 (Figure 25). Neither of the models could have net gain in 2016 – 2017 (Figure 26), but we can agree that the higher the k-value the better the result.

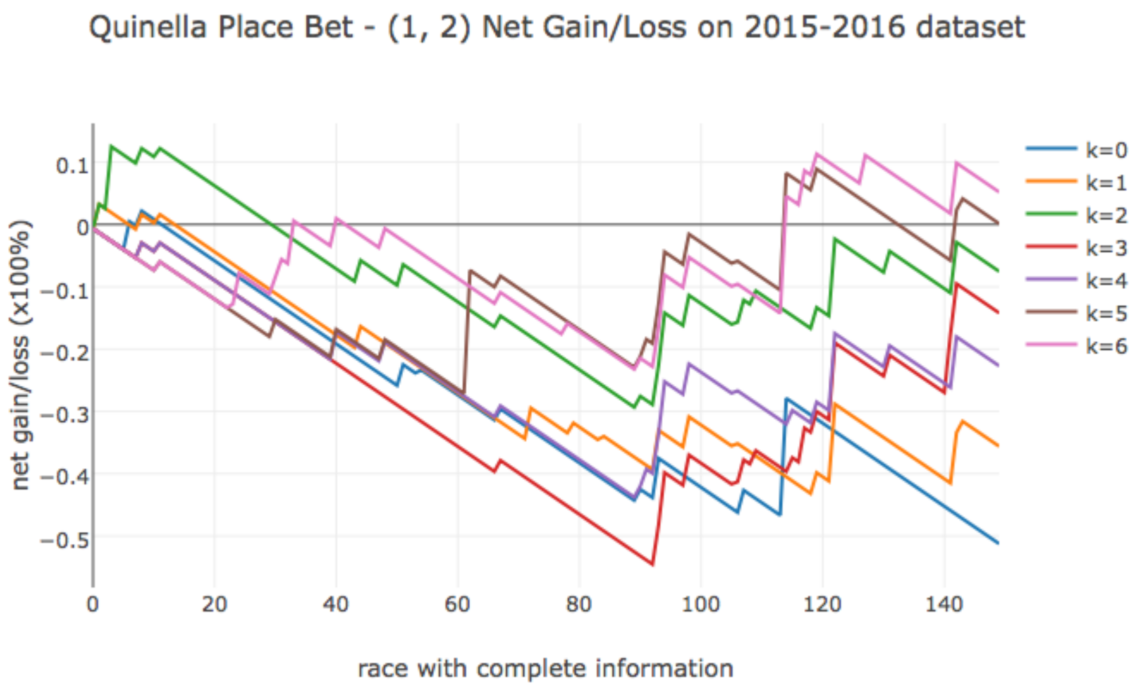


Figure 25

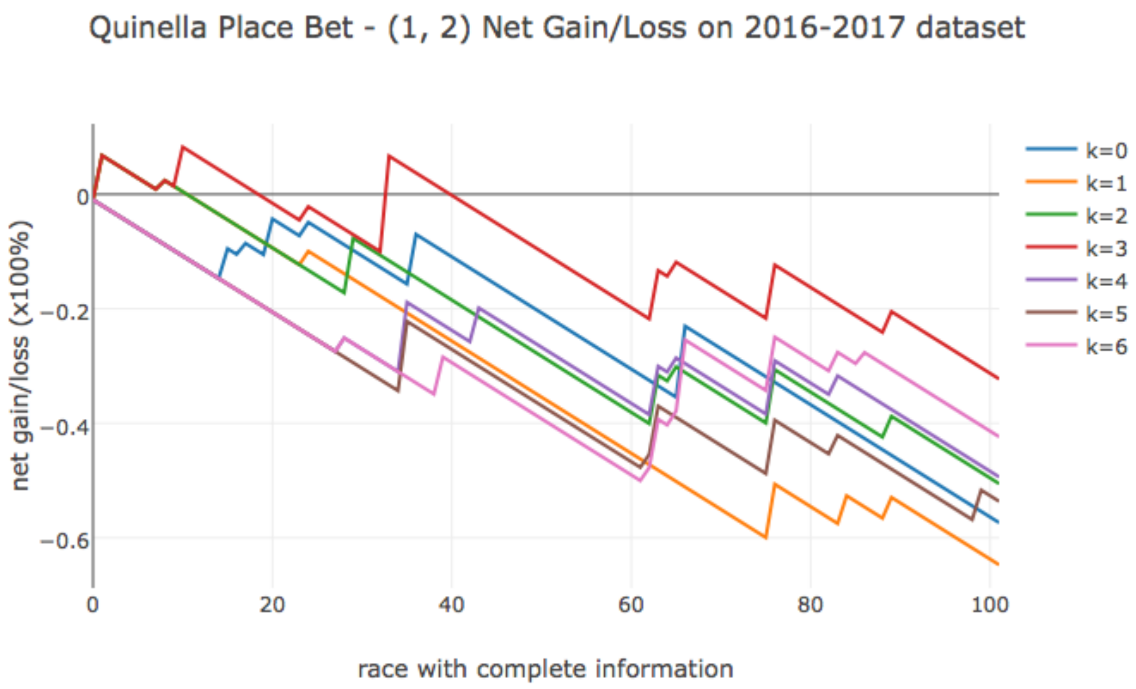

Quinella Place Bet - (1, 2) Net Gain/Loss on 2016-2017 dataset
net gain/loss (x100%)
0
−0.2
−0.4
−0.6
0
20
40
60
80
100
race with complete information
k=0
k=1
k=2
k=3
k=4
k=5
k=6

Figure 26

Quinella Place Bet – (1, 3)

The result of Quinella Place Bet – (1, 3) are a little bit wired. In 2015 – 2016 (Figure 27), the results are quite standard, the model with parameter k equals to 4 and 6 could have net gain. However, the only model could have net gain in 2016 – 2017 is the model with parameter k equals to 1, it is violating the agreement of "the higher the k-value, the better the result".

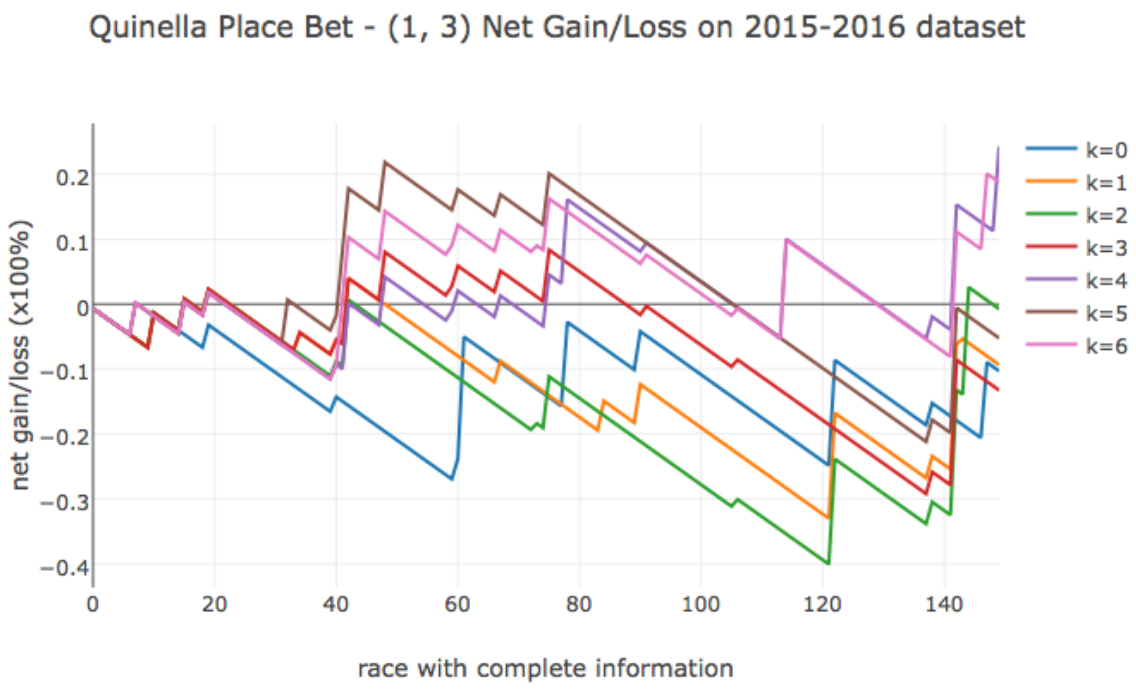


Figure 27

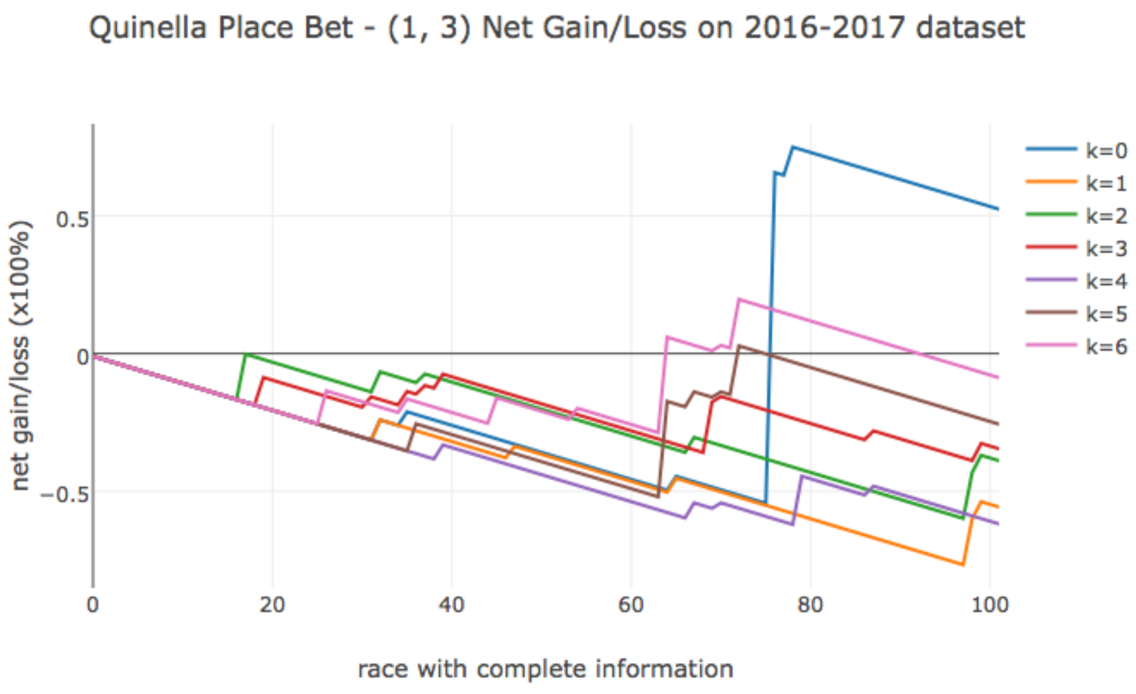

Quinella Place Bet - (1, 3) Net Gain/Loss on 2016-2017 dataset
net gain/loss (x100%)
0.5
0
−0.5
0
20
40
60
80
100
race with complete information
k=0
k=1
k=2
k=3
k=4
k=5
k=6

Figure 28

Quinella Place Bet – (2, 3)

The models perform better on Quinella Place Bet – (2, 3) comparing with the result of Quinella Place Bet – (1, 3). The models with parameter k equals to 1, 2, 3 and 4 could generate net gain in 2015 – 2016 (Figure 29); the models with parameter k equals to 1, 2, 5 could generate net gain in 2016 – 2017 (Figure 30); models with parameter k equals to 6 did a bad job among the years.

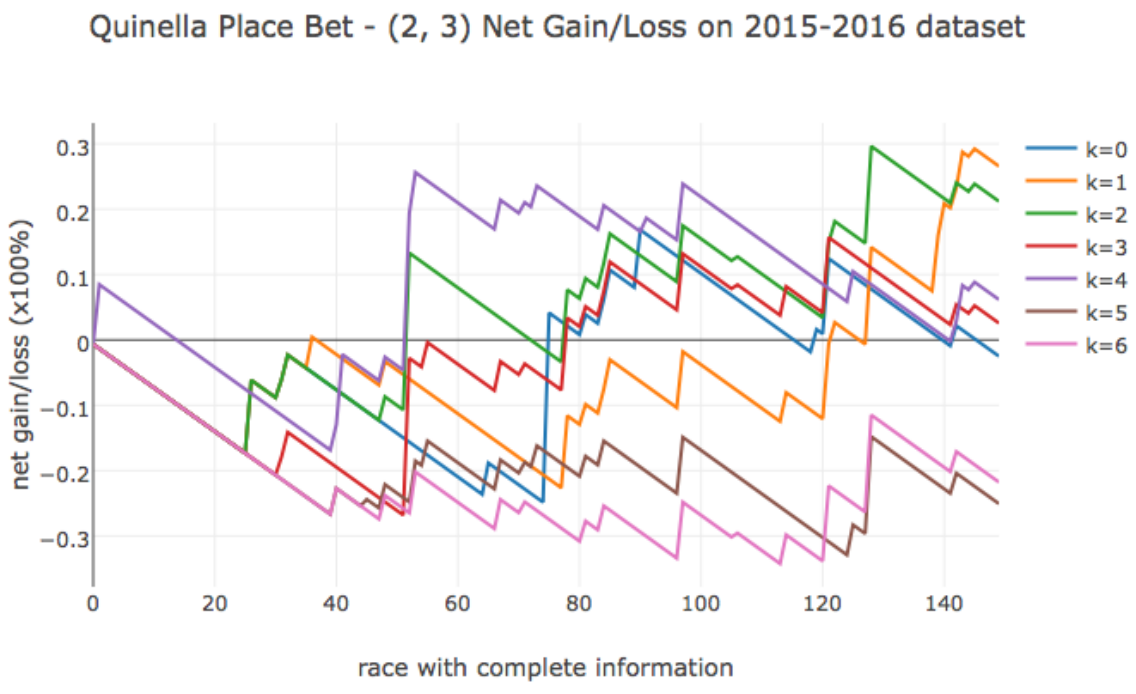


Figure 29

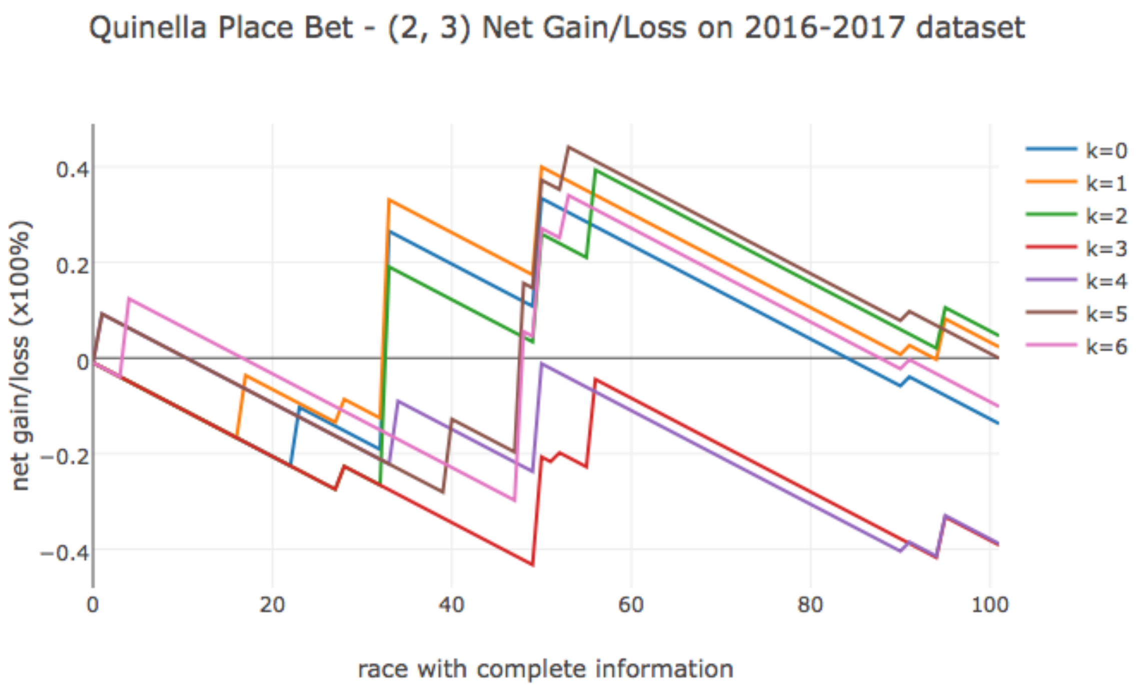

Quinella Place Bet - (2, 3) Net Gain/Loss on 2016-2017 dataset
net gain/loss (x100%)
0.4
0.2
0
−0.2
−0.4
0
20
40
60
80
100
race with complete information
k=0
k=1
k=2
k=3
k=4
k=5
k=6

Figure 30

Performance

We can look at the general performance among models by constructing a heat map. Regarding the color, more red means higher the net gain, more blue means the lower the net gain. In general, we can agree that the model with a higher parameter K trend to yield a higher net gain. In 2015 – 2016 and 2016 – 2017, the model with the parameter K equals to 6 has good net gain across different types of betting.

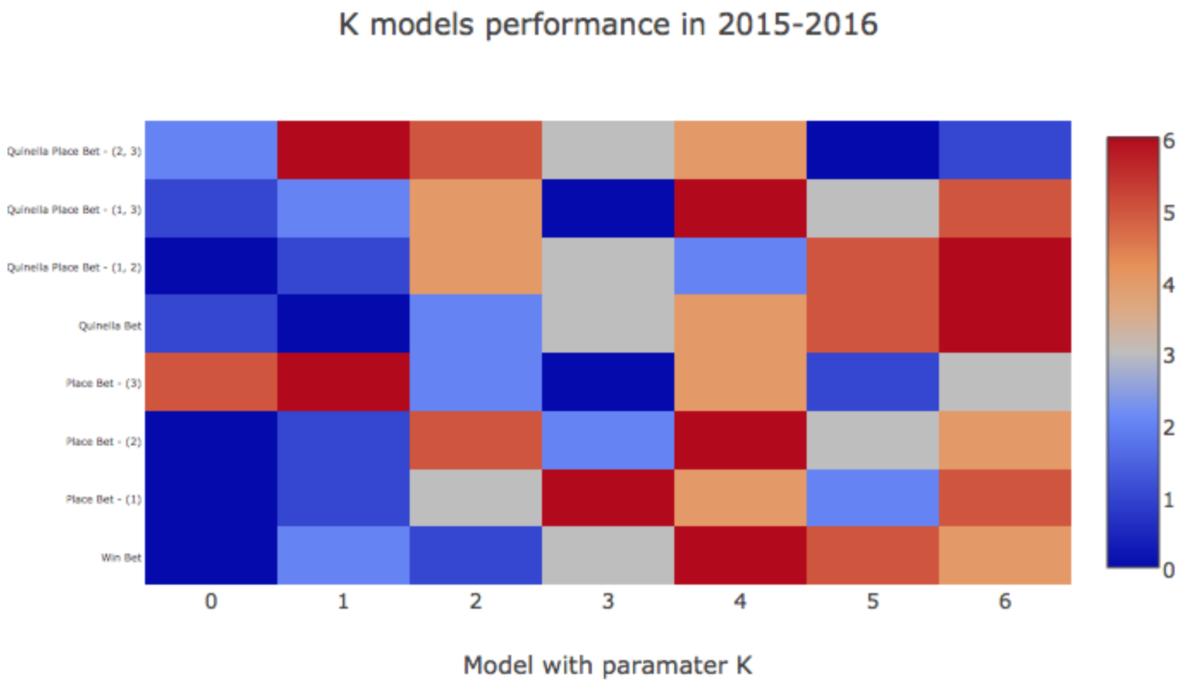


Figure 31

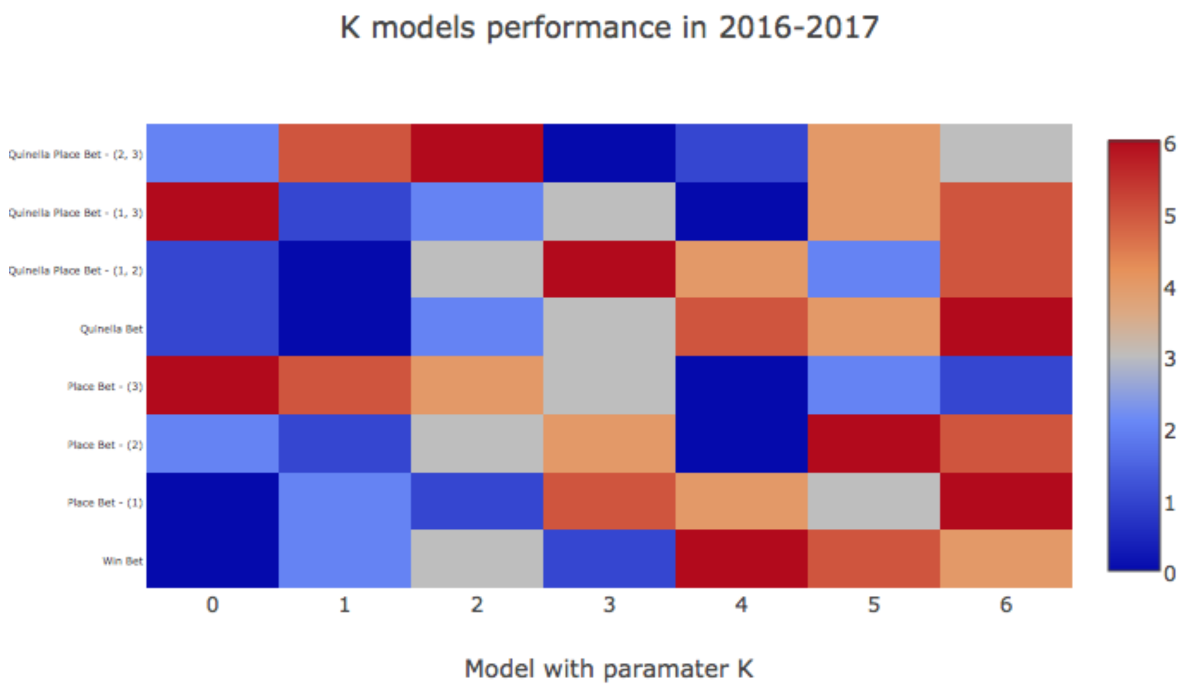

K models performance in 2016-2017
Quinella Place Bet - (2, 3)
Quinella Place Bet - (1, 3)
Quinella Place Bet - (1, 2)
Quinella Bet
Place Bet - (3)
Place Bet - (2)
Place Bet - (1)
Win Bet
0
1
2
3
4
5
6
Model with paramater K
6
5
4
3
2
1
0

Figure 32

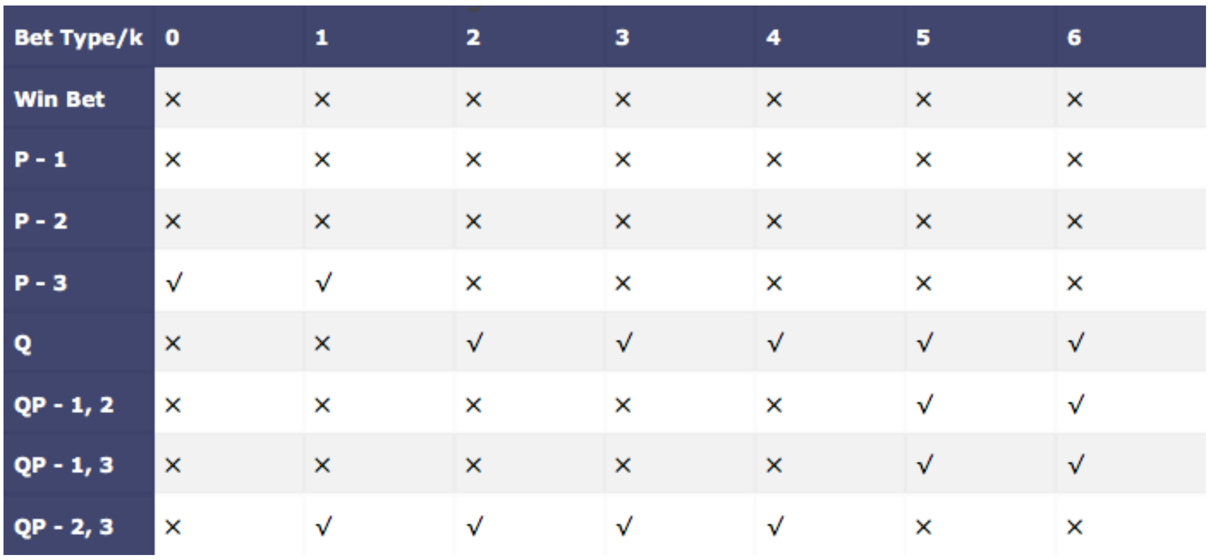
Over all Result

In general, the model with parameter k equals to 6 has the best performance (Figure 31), and we observed that 6-past records model did very well and fair on Quinella Bet and Quinella Place Bet (1 ,2). Also The with parameter k equals to 6 performed really well, they have net gain on Quinella Bet in 2015 – 2016 (Figure 23) and 2016 – 2017 (Figure 24). Therefore, we tried to simulate the combination of (1, 2, 3) on Quinella Bet (Figure 33), the result is good and stable in 2015-2016 and 2016-2017, though the net gain is lower than a single Quinella Bet (1, 2).

2015 – 2016

In 2015 – 2016, some of the models could generate net gain in some types of betting (Table 9). In general, we can generally agree that most of the models perform well on Quinella Bet.

Table 9

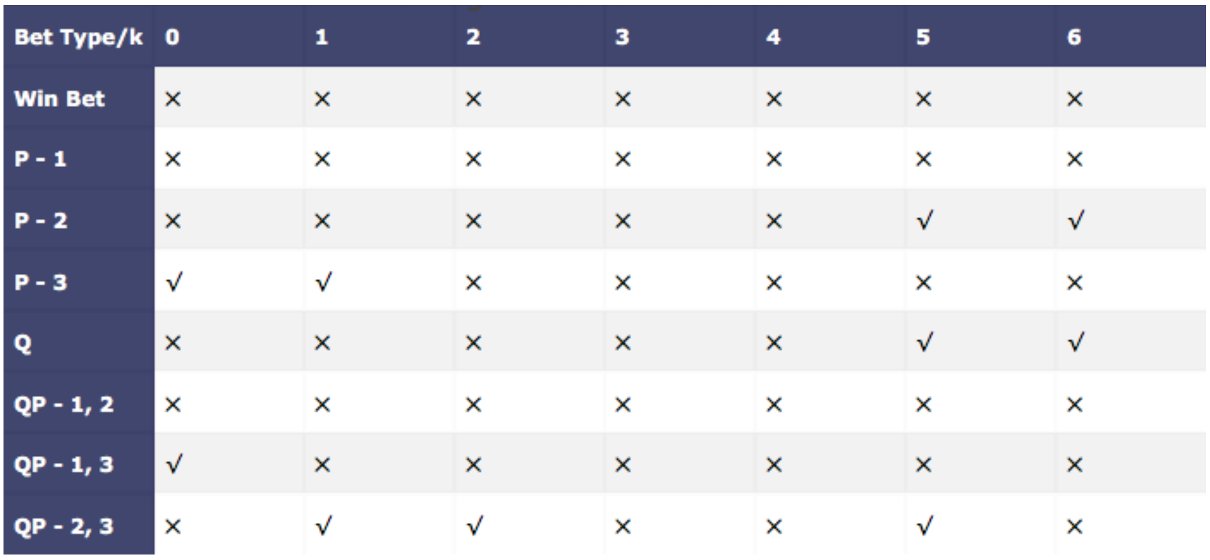


Bet Type/k	0	1	2	3	4	5	6
Win Bet	×	×	×	×	×	×	×
P - 1	×	×	×	×	×	×	×
P - 2	×	×	×	×	×	×	×
P - 3	√	√	×	×	×	×	×
Q	×	×	√	√	√	√	√
QP - 1, 2	×	×	×	×	×	√	√
QP - 1, 3	×	×	×	×	×	√	√
QP - 2, 3	×	√	√	√	√	×	×

2016 – 2017

In 2016 – 2017, the models were not performed as well as the result in 2015 – 2016. However, the models with parameter k equals to 5 and 6 were doing well on Quinella Bet, which are similar results of 2015 – 2016.

Table 10



Bet Type/k	0	1	2	3	4	5	6
Win Bet	×	×	×	×	×	×	×
P - 1	×	×	×	×	×	×	×
P - 2	×	×	×	×	×	√	√
P - 3	√	√	×	×	×	×	×
Q	×	×	×	×	×	√	√
QP - 1, 2	×	×	×	×	×	×	×
QP - 1, 3	√	×	×	×	×	×	×
QP - 2, 3	×	√	√	×	×	√	×

Combination Bet

Quinella Bet – (1,2) (1,3) (2,3)

Almost every models performed really well on Quinella Bet. Only the model with parameter k equals to 4 could not generate net gain in 2015 – 2016 (Figure 33). However, the result in 2016 – 2017 is quite disappointing, only the model with parameter k equals to 6 have net gain.

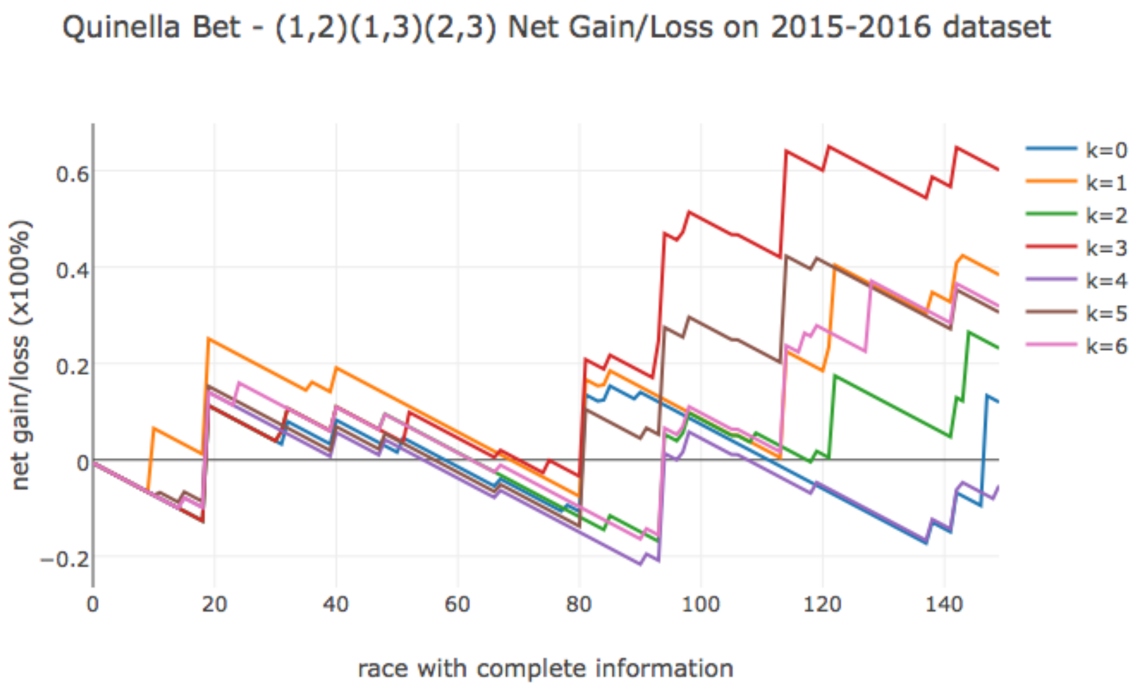


Figure 33

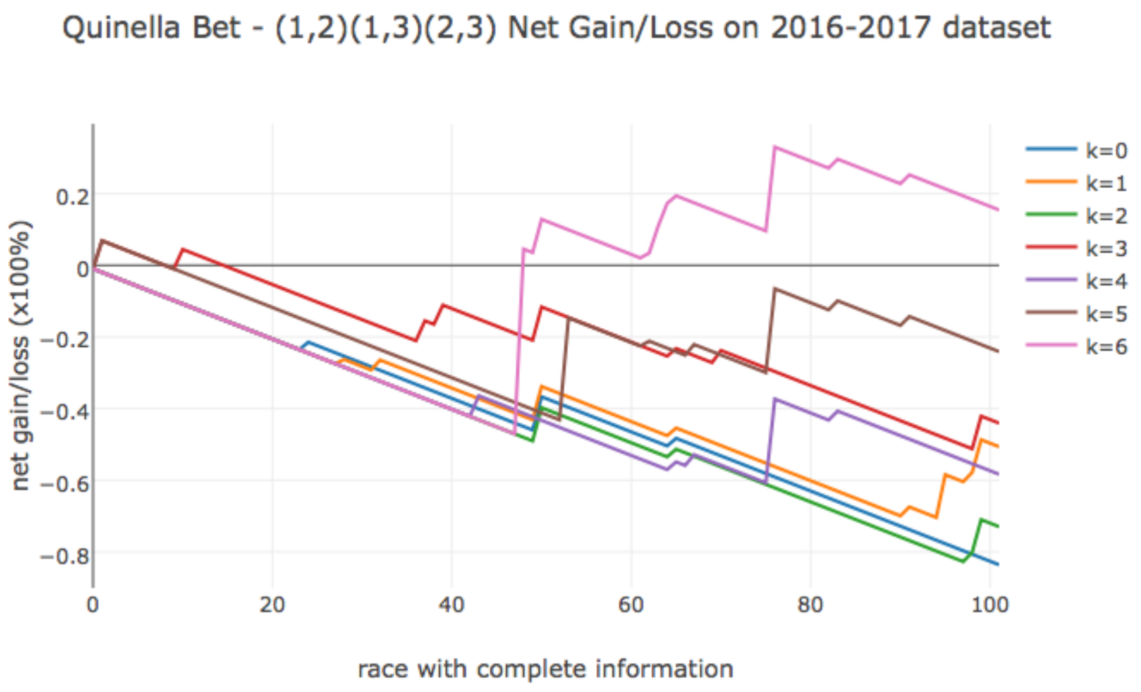

Quinella Bet - (1,2)(1,3)(2,3) Net Gain/Loss on 2016-2017 dataset
net gain/loss (x100%)
0.2
0
−0.2
−0.4
−0.6
−0.8
0
20
40
60
80
100
race with complete information
k=0
k=1
k=2
k=3
k=4
k=5
k=6

Figure 34

Special Condition

Definition

After the simulation, we further study the predicted results. We found that the models do particularly well under a special condition. Define $Predicted(x)$ as a function of x, where x is a row vector, the function will output the predicted time of the instance x. Recall that we had defined $r_{\hat{r}}^{(i)}$ be the row vector corresponding to the i fastest horse in a race with identity $\hat{r}$. Let α be the absolute different of the predicted time of the top 2 horses predicted by our model.

$$\alpha = abs(Predicted(r_{\hat{r}}^{(1)}) - Predicted(r_{\hat{r}}^{(2)}))$$

When α is smaller than a small value, namely ε. The models do so well under this condition.

$$\alpha < \varepsilon$$

So we decided to play a race if and only if the race satisficed the above condition. However, what is the appropriate ε value?

Normal Bet

Win Bet (Under Special Condition)

We can see that no matter what sort of threshold we have set, neither of our model could generate net gain in 2015 – 2016 and 2016 – 2017, as our models did not perform well on win bet. Therefore, it is making sense that no red color in the contour map.

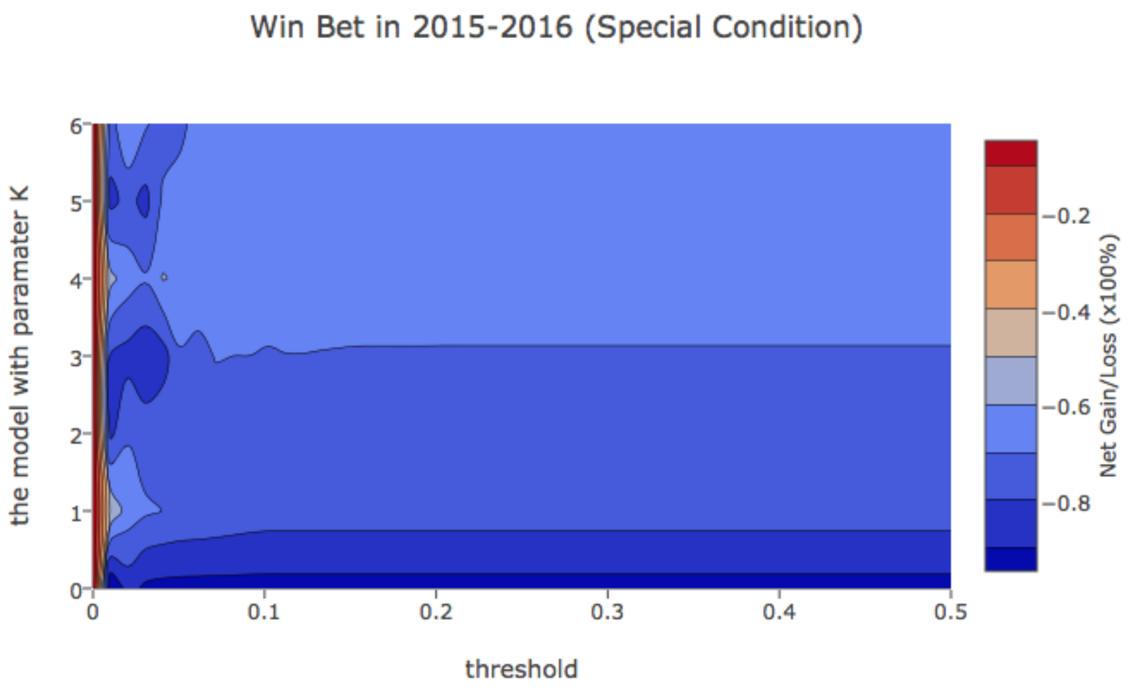


Figure 35

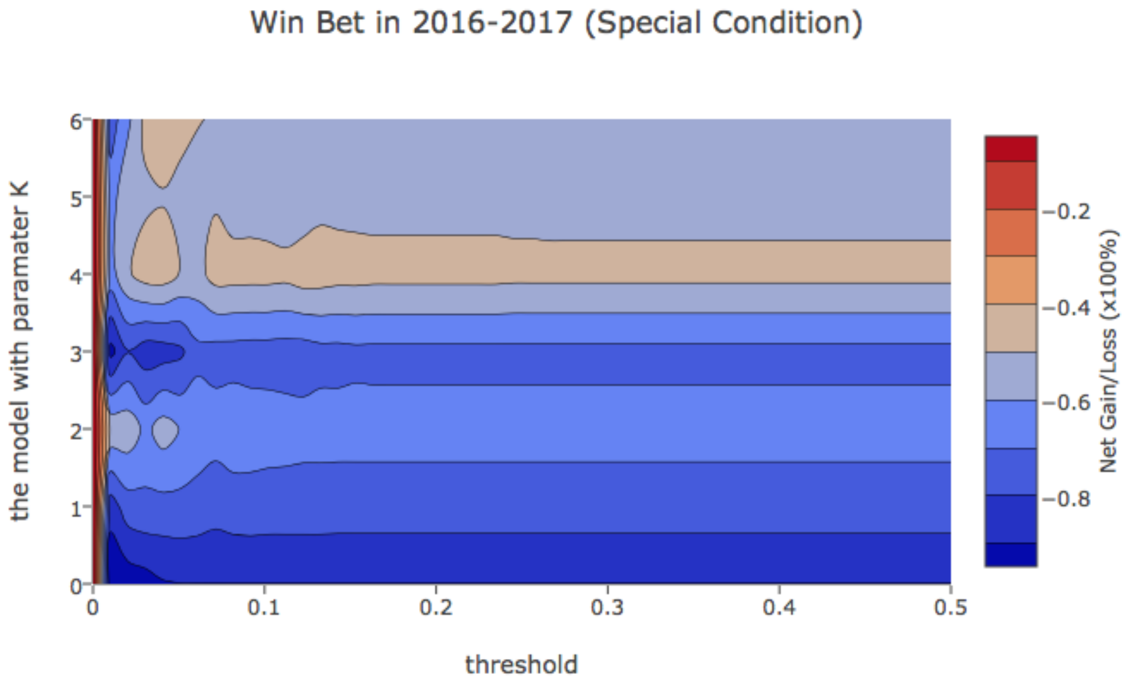

Win Bet in 2016-2017 (Special Condition)
the model with paramater K
threshold
Net Gain/Loss (x100%)
0
1
2
3
4
5
6
0
0.1
0.2
0.3
0.4
0.5
−0.2
−0.4
−0.6
−0.8

Figure 36

Place Bet – (1) (Special Condition)

In place bet – (1), a model with parameter k equals to 4 and with the threshold around 0.03 were able to achieve possible net profit in 2015 – 2016. We can also see that the model with parameter k equals to 6 and with the threshold around 0.04 could also yield a fair result. In 2016 – 2017, the models with parameter k greater than 3 could generate net gain in general, no matter what threshold was set, though the gain is small.

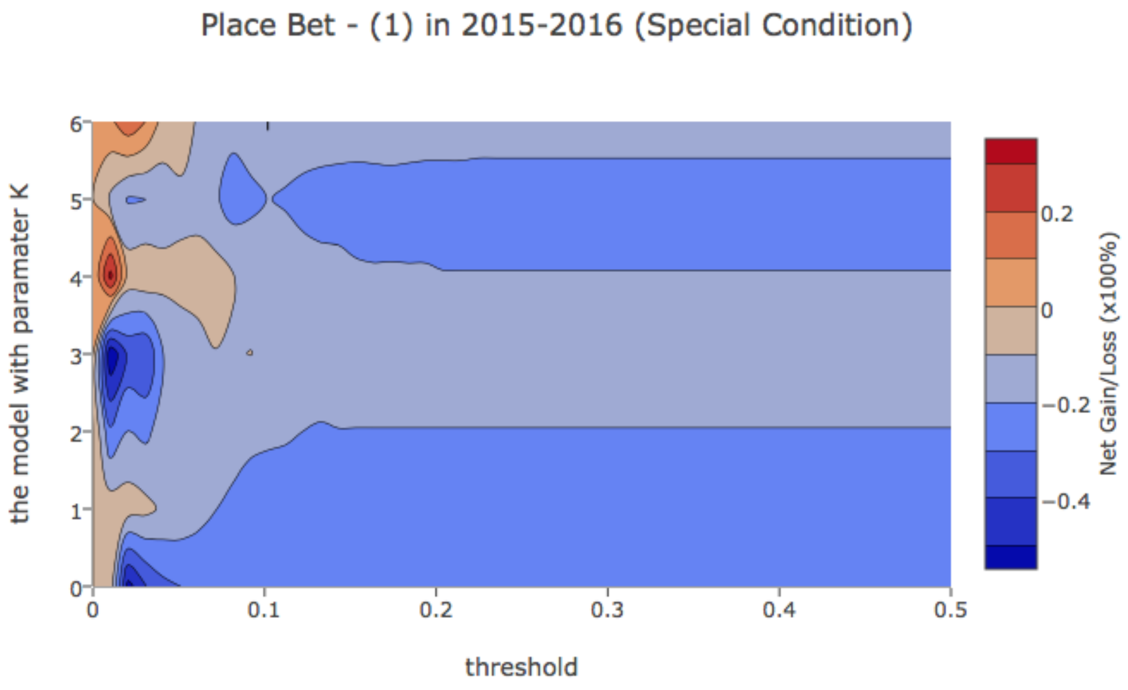
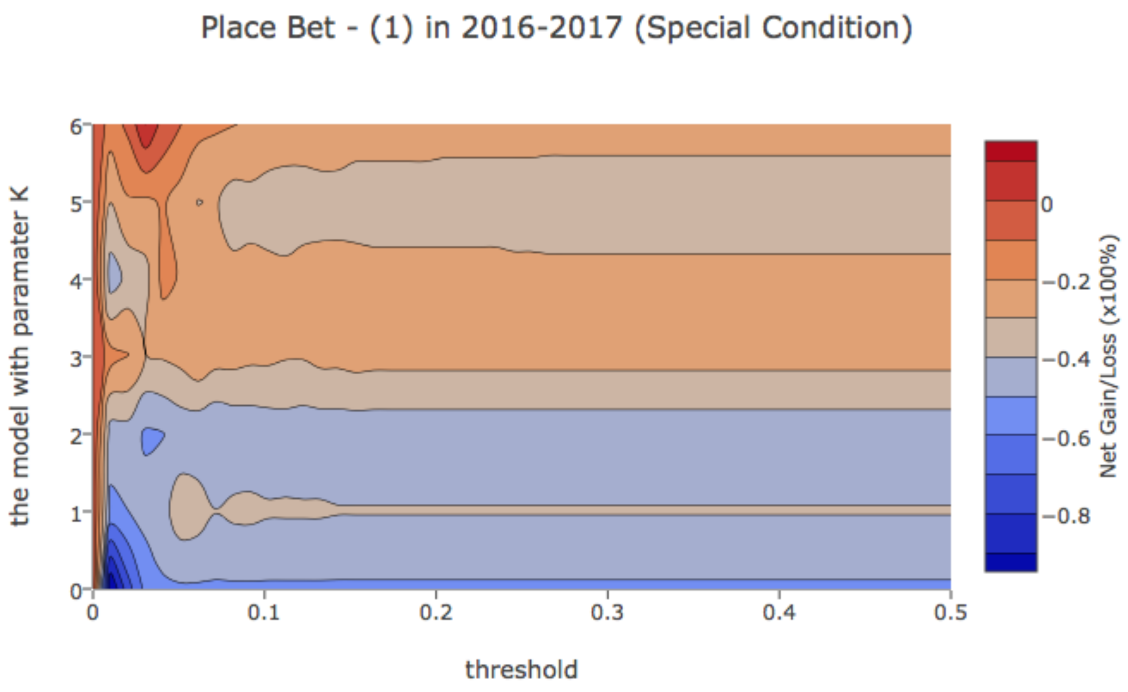


Figure 37



Place Bet - (1) in 2016-2017 (Special Condition)
the model with paramater K
0
1
2
3
4
5
6
threshold
0
0.1
0.2
0.3
0.4
0.5
Net Gain/Loss (x100%)
0
−0.2
−0.4
−0.6
−0.8

Figure 38

Place Bet – (2) (Under Special Condition)

In Place Bet – (2) under special condition, models with parameter k equals to 4, 5 and 6 with the threshold around 0.03 to 0.05, could generate net gain in 2015 – 2016. In general, threshold in between 0 and 0.1 for k ranging from 4 to 6 could generate profit in 2015 - 2016. The result is much better in 2016 – 2017, we can agree that the high k value with a appropriate threshold could generate profit.

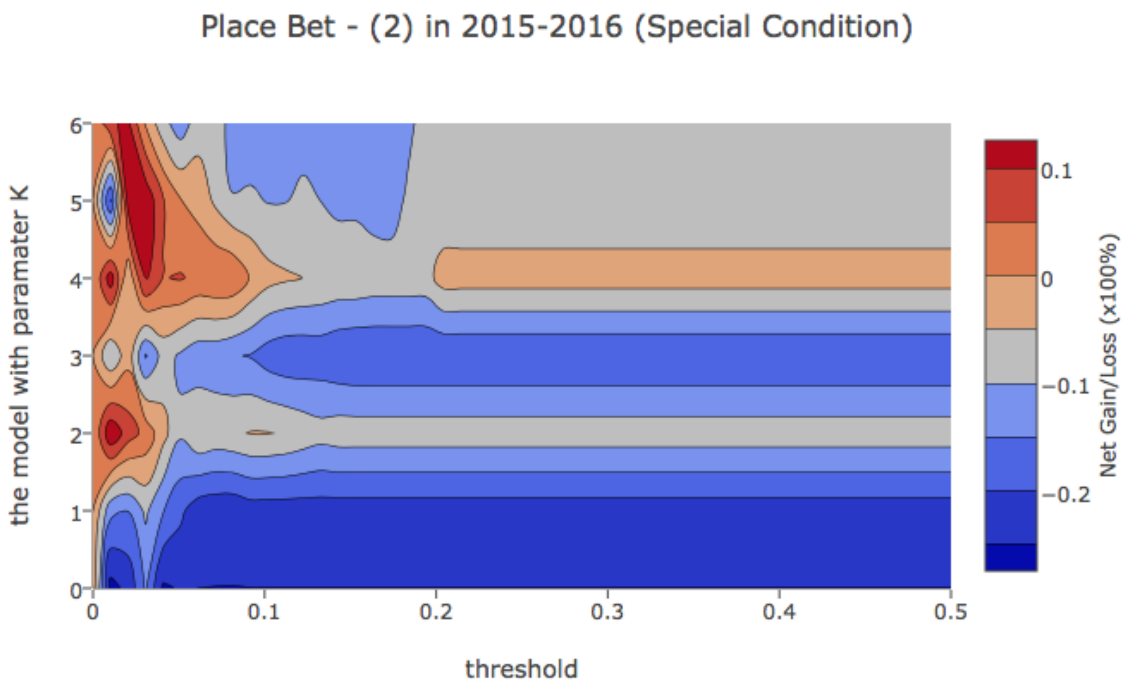
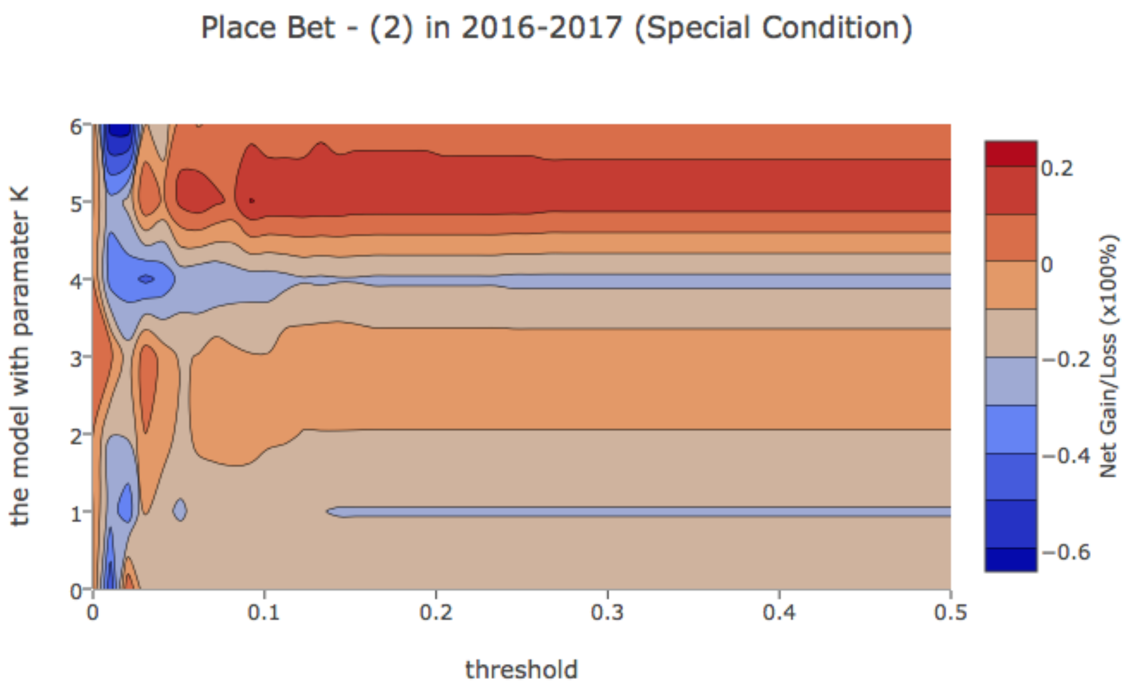


Figure 39



Place Bet - (2) in 2016-2017 (Special Condition)
the model with paramater K
threshold
Net Gain/Loss (x100%)
0
1
2
3
4
5
6
0.1
0.2
0.3
0.4
0.5
−0.6
−0.4
−0.2
0.2

Figure 40

Place Bet – (3) (Under Special Condition)

In Place Bet – (3) under special condition, the models could generate net profit with the threshold from 0 to 0.5, and it is wired that the model with parameter k equals to 1 with the threshold around 0.03 turns out performed the best in 2015 – 2016. In 2016 – 2017, the model with parameter k equals to 6 with threshold ranging from 0.02 to 0.03 performed the best.

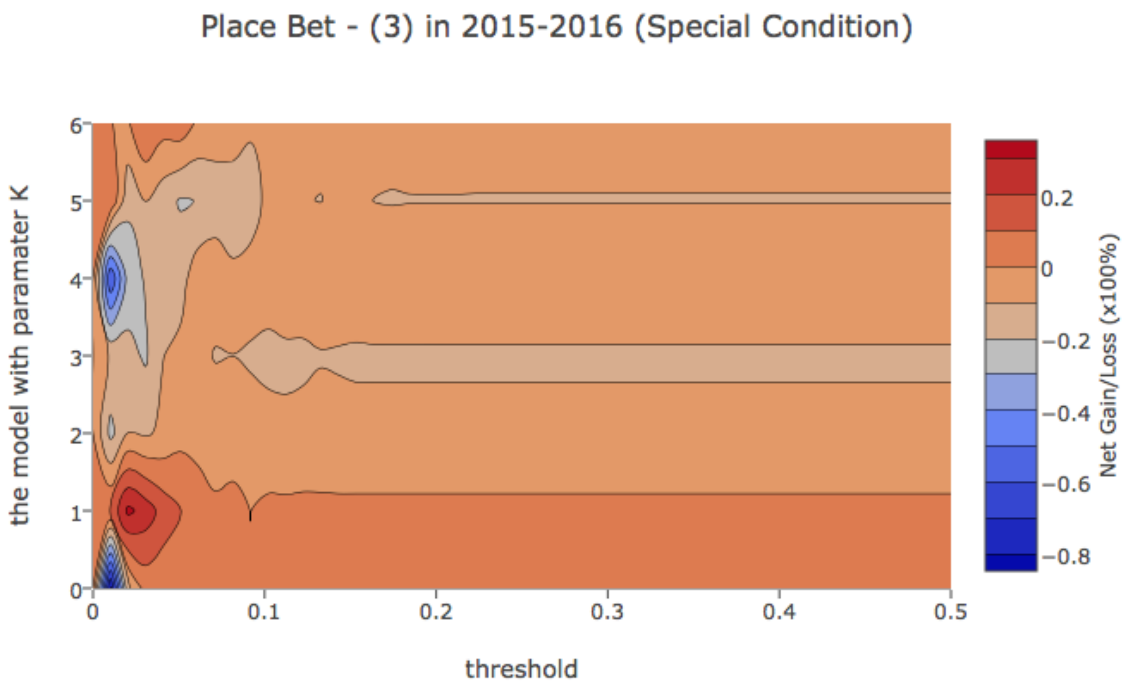


Figure 41

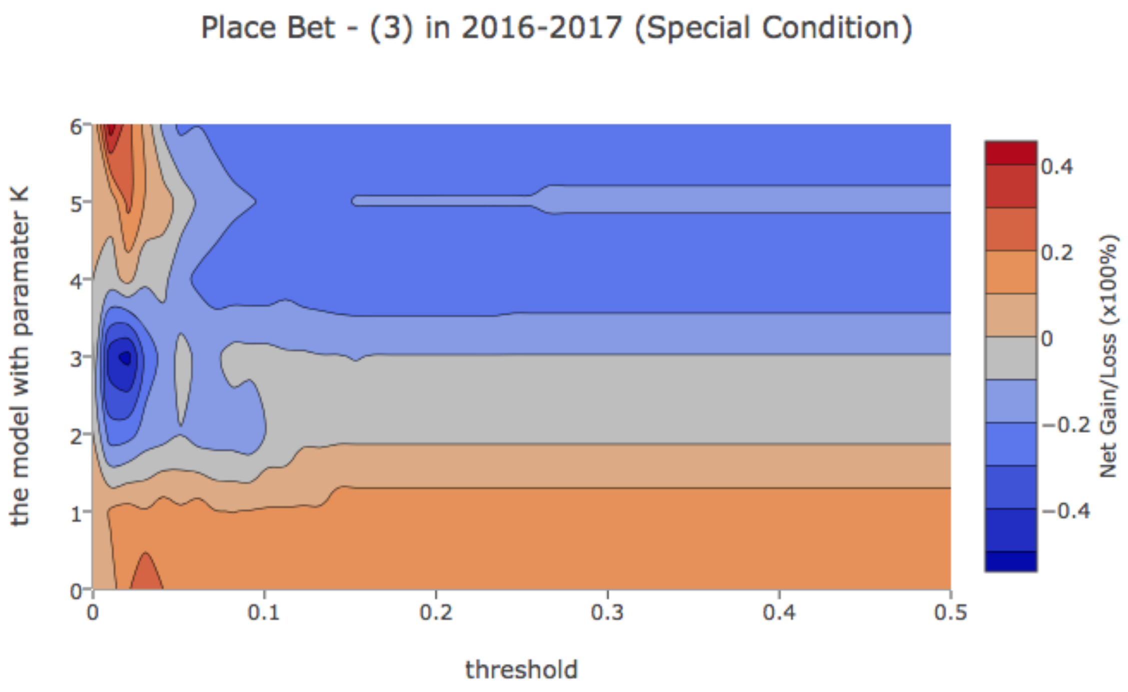

Place Bet - (3) in 2016-2017 (Special Condition)
the model with paramater K
threshold
Net Gain/Loss (x100%)
0
1
2
3
4
5
6
0
0.1
0.2
0.3
0.4
0.5
0.4
0.2
0
−0.2
−0.4

Figure 42

Quinella Bet – (Under Special Condition)

In Quinella Bet under special condition, the models with parameter k equals to 5 and 6 with threshold around 0.03 performed really well, the net gain was over 200% in 2015 -2016. The result in 2016 – 2017 is similar to the result of 2015 – 2016, the model with parameter k equals to 6 with threshold 0.03 could also generate net gain, though not as much as in 2015 – 2016.

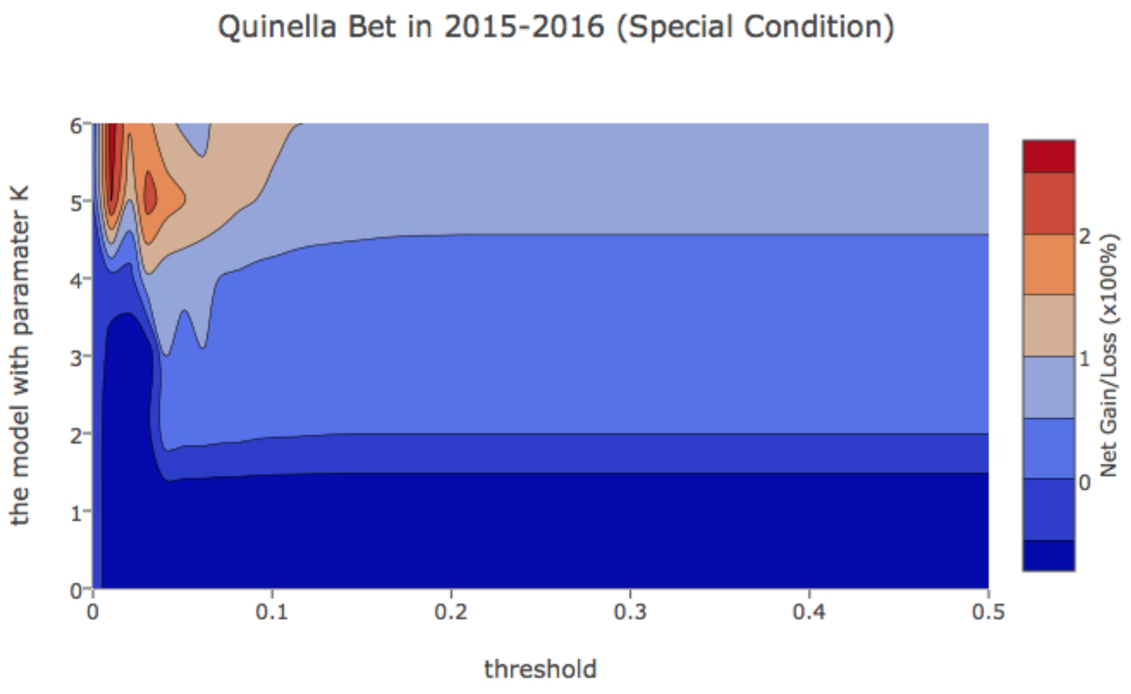


Figure 43

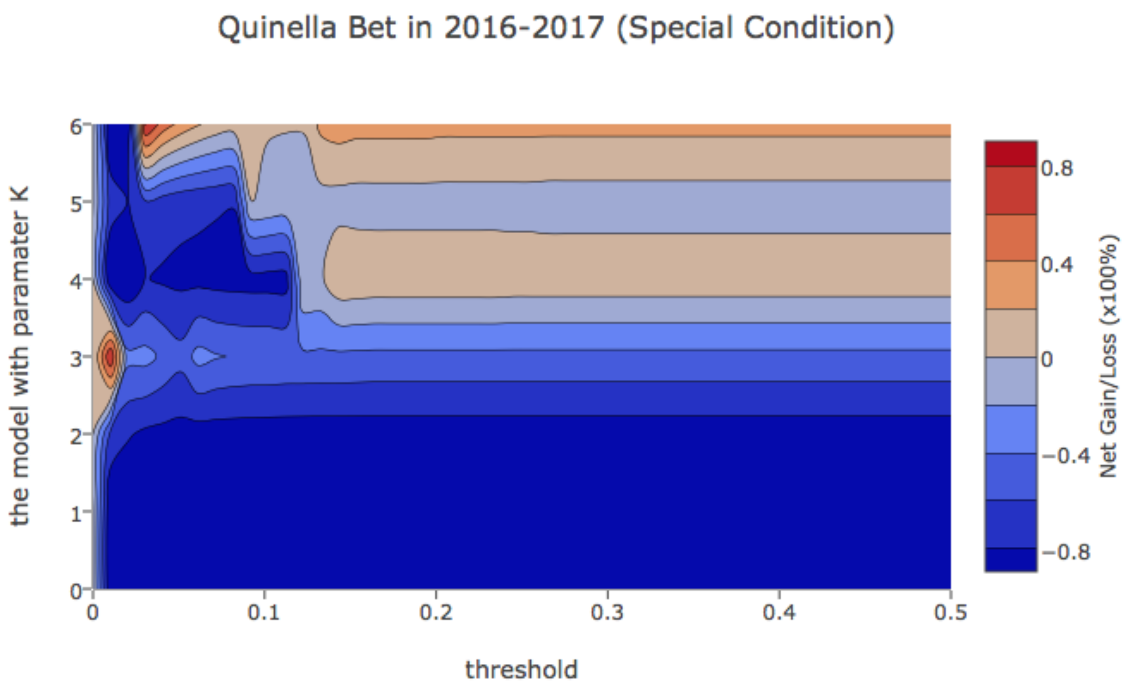

Quinella Bet in 2016-2017 (Special Condition)
the model with paramater K
threshold
Net Gain/Loss (x100%)
0
1
2
3
4
5
6
0
0.1
0.2
0.3
0.4
0.5
0.8
0.4
0
−0.4
−0.8

Figure 44

Quinella Place Bet – (1, 2) (Under Special Condition)

In the Quinella Place Bet (1, 2) under special condition, the model with parameter k equals to 6 with threshold around 0.01 – 0.02 performed really well in 2015 – 2016, while it performed badly in 2016 – 2017.

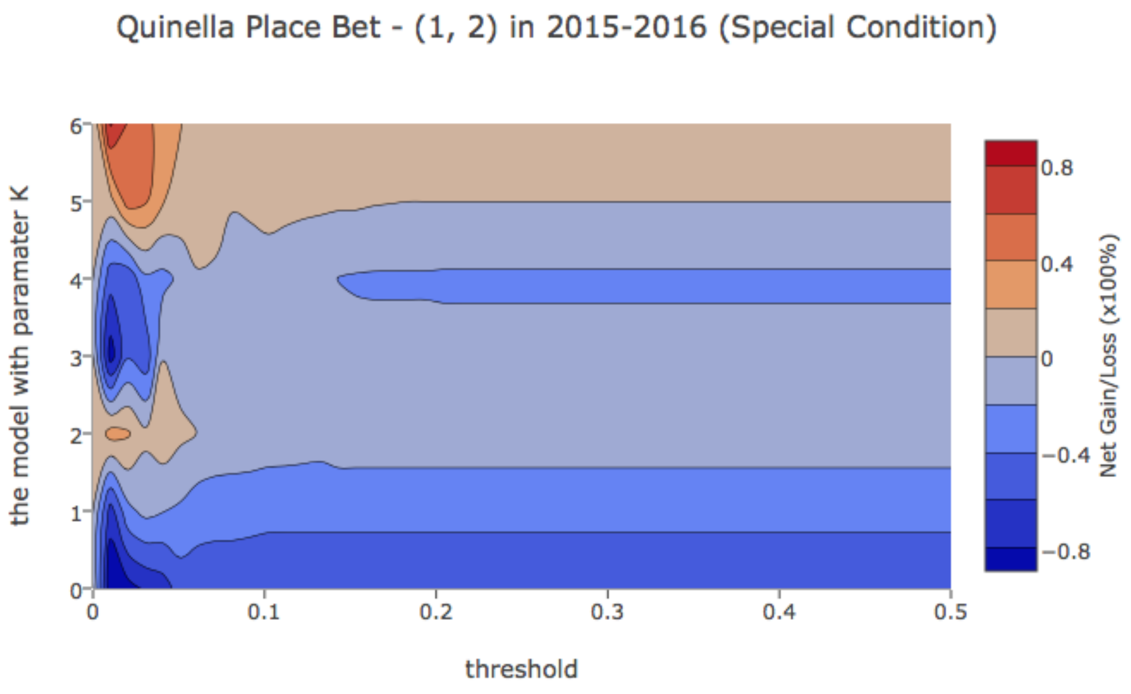


Figure 45

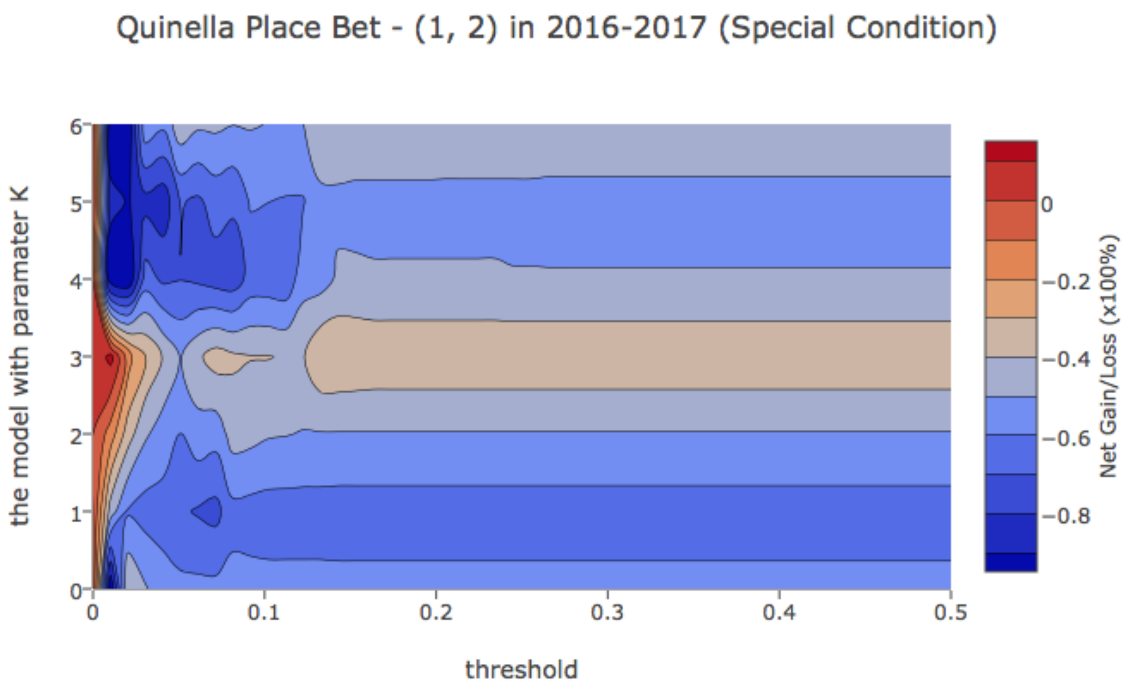

Quinella Place Bet - (1, 2) in 2016-2017 (Special Condition)
the model with paramater K
threshold
Net Gain/Loss (x100%)
0
−0.2
−0.4
−0.6
−0.8
0
1
2
3
4
5
6
0
0.1
0.2
0.3
0.4
0.5

Figure 46

Quinella Place Bet – (1, 3) (Under Special Condition)

In Quinella Place Bet – (1, 3) under special condition, the result of 2015 – 2016 is similar to the result of Quinella Place Bet – (1, 2) in 2015 – 2016, the model with parameter k equals to 6 with threshold ranging around 0.02 – 0.03 performed the best in 2015 – 2016, and the result is similar in 2016 – 2017.

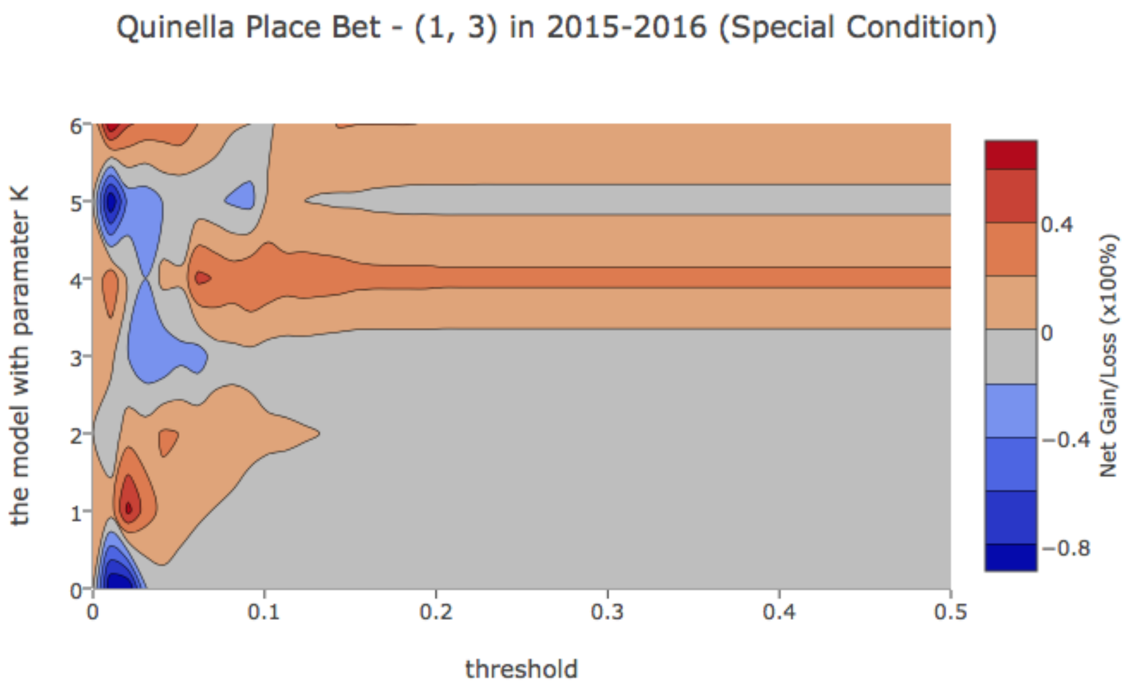


Figure 47

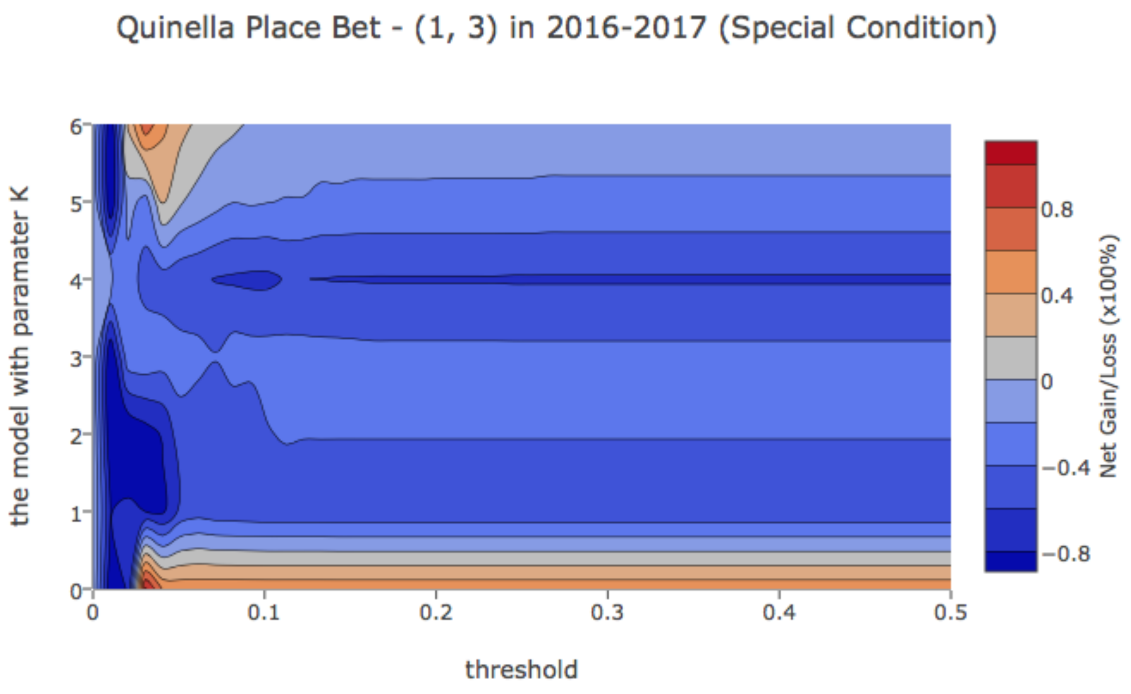

Quinella Place Bet - (1, 3) in 2016-2017 (Special Condition)
the model with paramater K
6
5
4
3
2
1
0
0
0.1
0.2
0.3
0.4
0.5
threshold
0.8
0.4
0
−0.4
−0.8
Net Gain/Loss (x100%)

Figure 48

Quinella Place Bet – (2, 3) (Under Special Condition)

In Quinella Place Bet – (2, 3) under special condition, the model with parameter k equals to 6 with threshold ranging around 0.02 – 0.03 performed the best in 2015 – 2016, while in 2016 – 2017, the models generally performed bad.

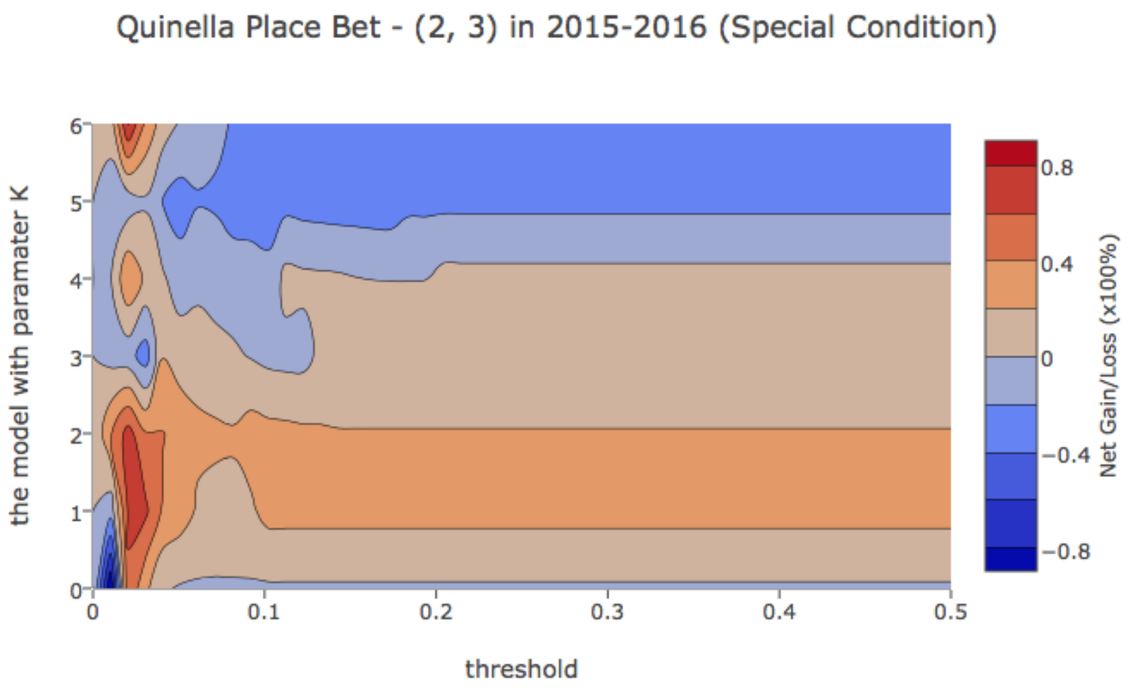


Figure 49

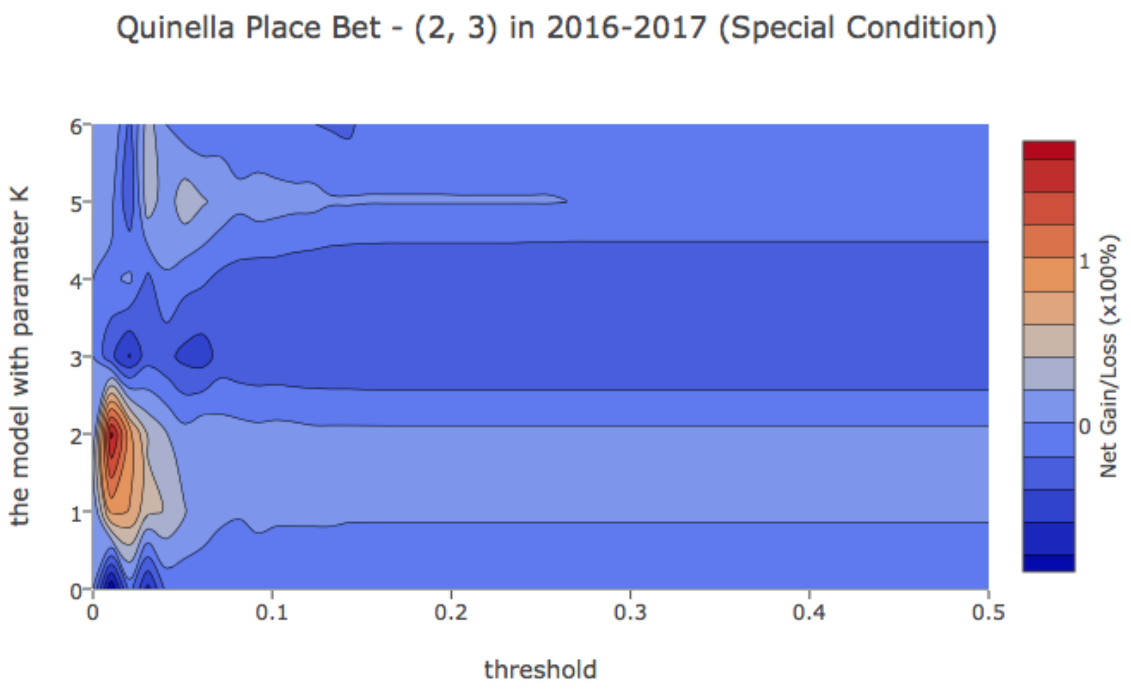

Quinella Place Bet - (2, 3) in 2016-2017 (Special Condition)
the model with paramater K
6
5
4
3
2
1
0
0
0.1
0.2
0.3
0.4
0.5
threshold
1
0
Net Gain/Loss (x100%)

Figure 50

Appropriate ε

We call $\alpha < \varepsilon$ as threshold. Since the threshold should be small, so we searched the threshold between the interval 0 to 0.5 on different kinds of betting. We could not find a threshold such that it could make a model gaining net profit on all kinds of betting. However, we observed that 0.03 is a good threshold, though it didn't allow the model to gain maximum profit on most of the betting type, it yields a fair or balance result in general. We can generally agree that the model with a high k-value and apply a low threshold will generally produce a good result. Our models did a good job on Quinella Bet, and a more stable result could be archived by betting on the combination of $\{1,2,3\}$ on Quinella Bet (Figure 51). So we decided to set the ε to 0.03.

Combination Bet

Quinella Combination Bet (Under Special Condition)

In Quinella Combination Bet under special condition, the models with parameter j equals to 6 and with threshold 0.03 performed really well in 2015 – 2016 and 2016 – 2017.

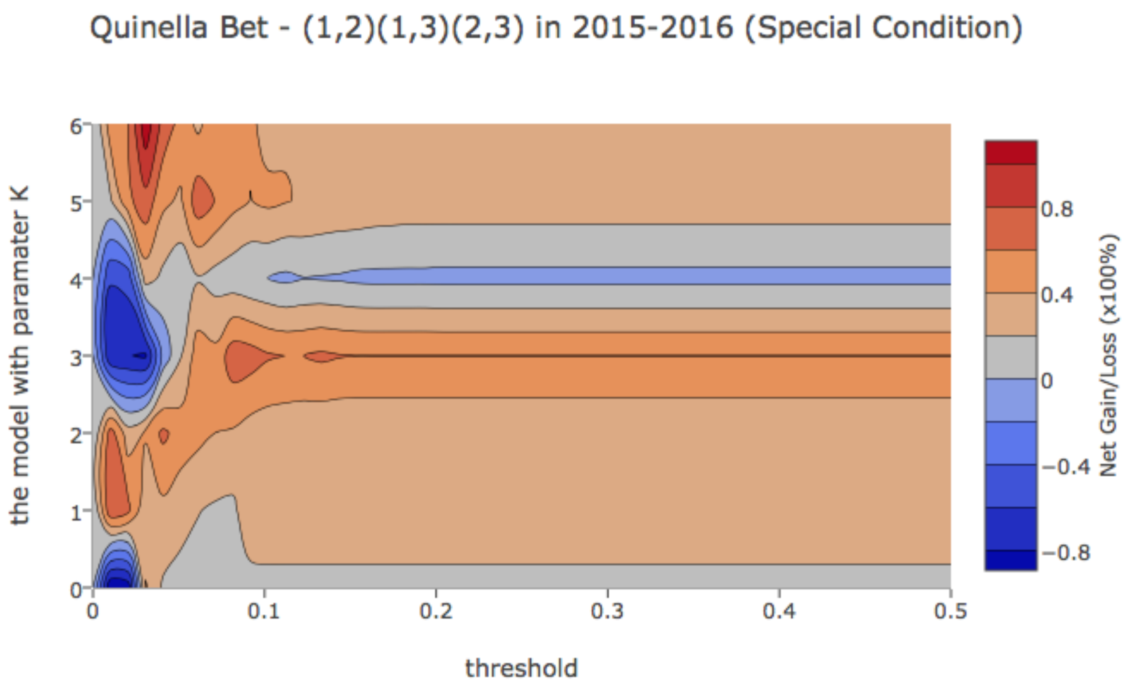
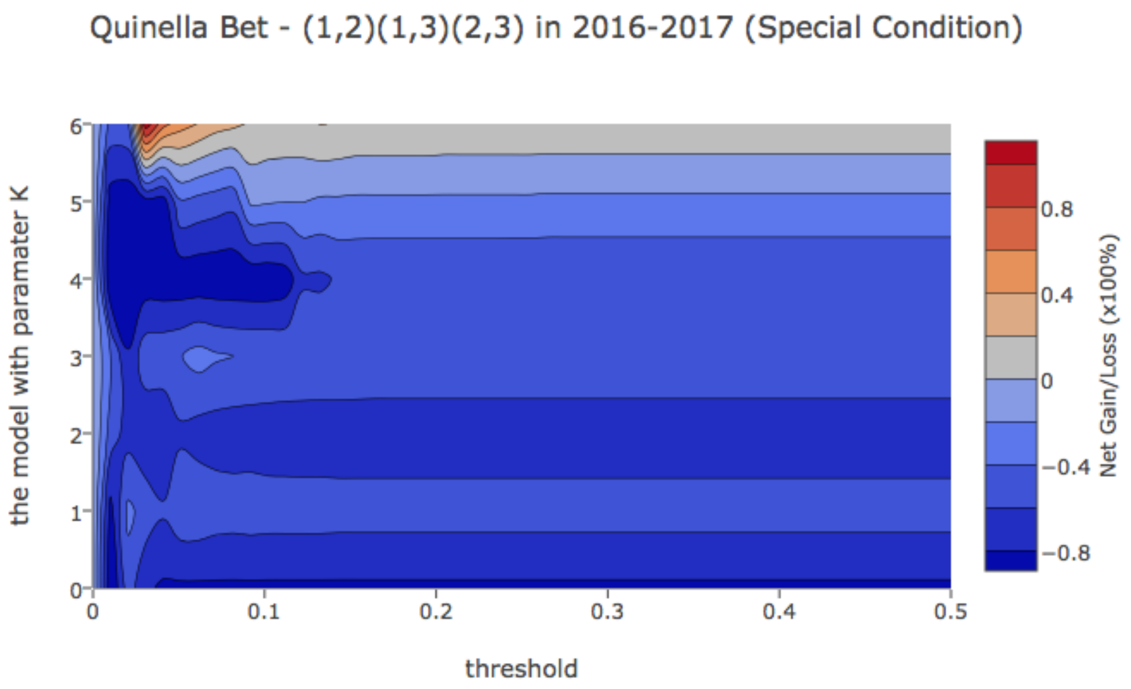


Figure 51



Quinella Bet - (1,2)(1,3)(2,3) in 2016-2017 (Special Condition)
the model with paramater K
0
1
2
3
4
5
6
threshold
0
0.1
0.2
0.3
0.4
0.5
Net Gain/Loss (x100%)
0.8
0.4
0
−0.4
−0.8

Figure 52

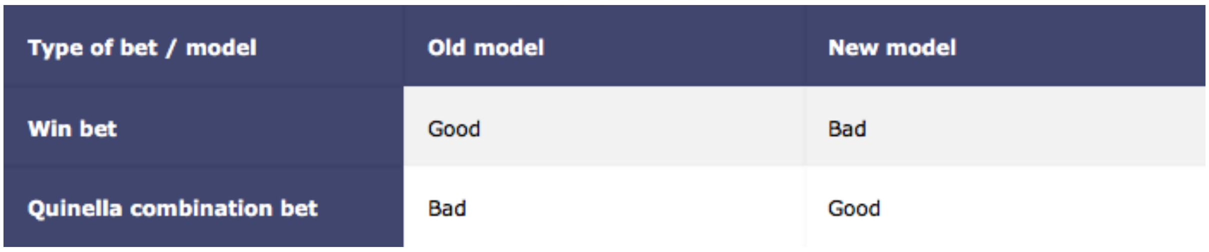
Compare with old model

Last semester, we have trained one deep neural network model and one linear classification model. Unfortunately, we were failed to produce the same result driven by the deep neural network after the Tensorflow was updated. We are going to compare our current model with the linear classification model from the last semester only.

Regarding the comparison methods, we are going to compare the net gain of playing win bet and the net gain of playing quinella combination bet. The testing dataset is 2015 – 2016. We cannot compare them in the same graph since we didn't filter any race record in the method that had been using in the last semester, so we will compare them separately.

Generally speaking, old model performed better on win bet and the new model do better on quinella combination bet $\{1,2,3\}$ (Table 11). Somehow the result is making sense, it is because the old model is a classification model, which is determining whether a particular horse will win, which means the first place, or not, so it makes sense that the old model did well on win bet. On the contrary, the new model is a regression model, which is predicting the finishing time of a particular horse in a race, it turns out the model will be good at ranking horses in a race; for instance, suggesting top 3 horses in a race. Therefore, it makes sense that the new model performed better than the old one on quinella combination bet $\{1,2,3\}$.

Table 11



Type of bet / model	Old model	New model
Win bet	Good	Bad
Quinella combination bet	Bad	Good

Win bet

In terms of win bet, old model with 75% threshold outperformed the new model. Old model with 75% threshold managed to generate net gain (Figure 54), and the new model loss around 70% of the money about one year (Figure 53).

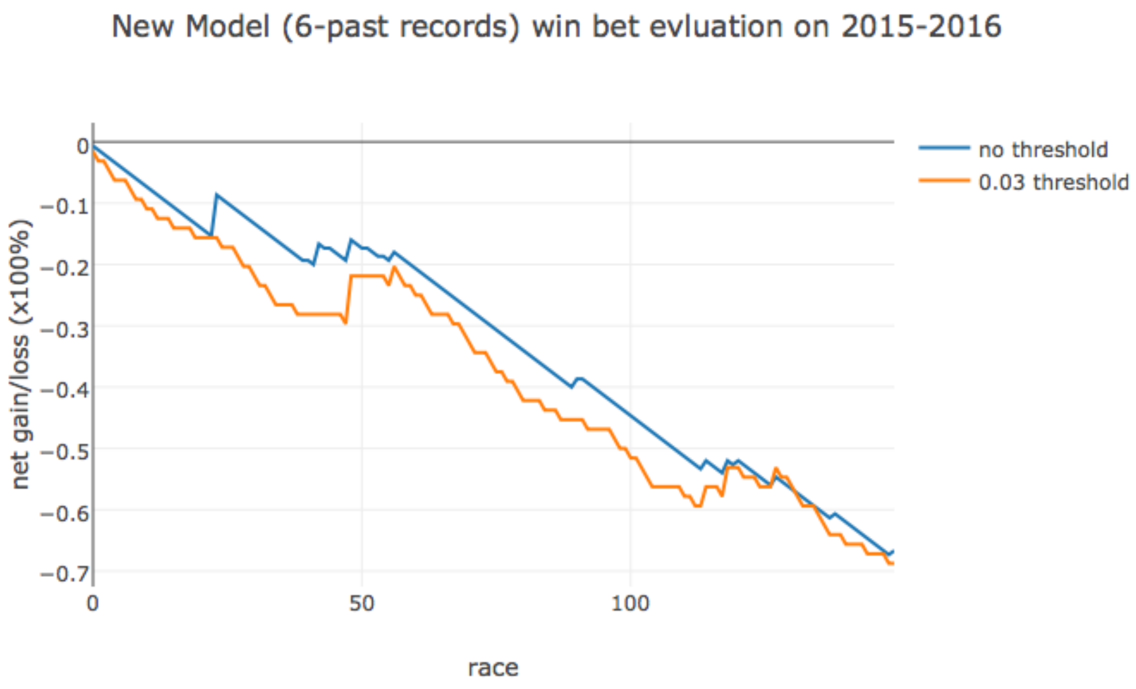


Figure 53

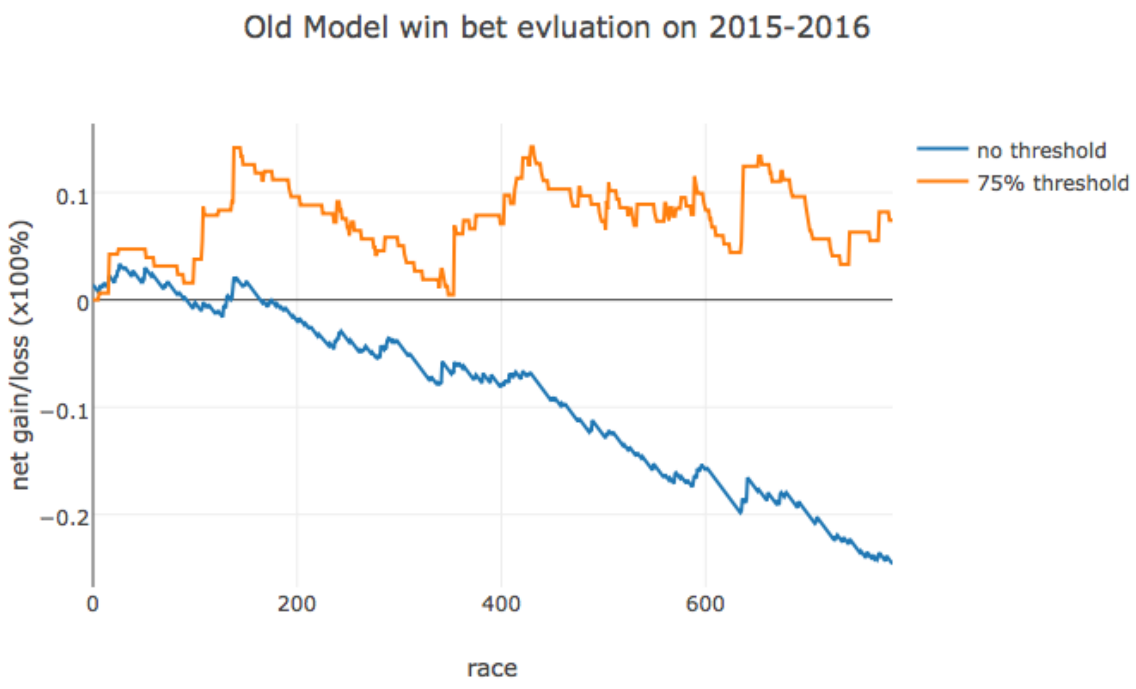

Old Model win bet evluation on 2015-2016
net gain/loss (x100%)
0.1
0
−0.1
−0.2
0
200
400
600
race
no threshold
75% threshold

Figure 54

Quinella Bet

In terms of quinella combination bet $\{1,2,3\}$, both of the new models and old models managed to generate net gain, and the new model with $\epsilon = 0.03$ generated over 110% net gain after a year.

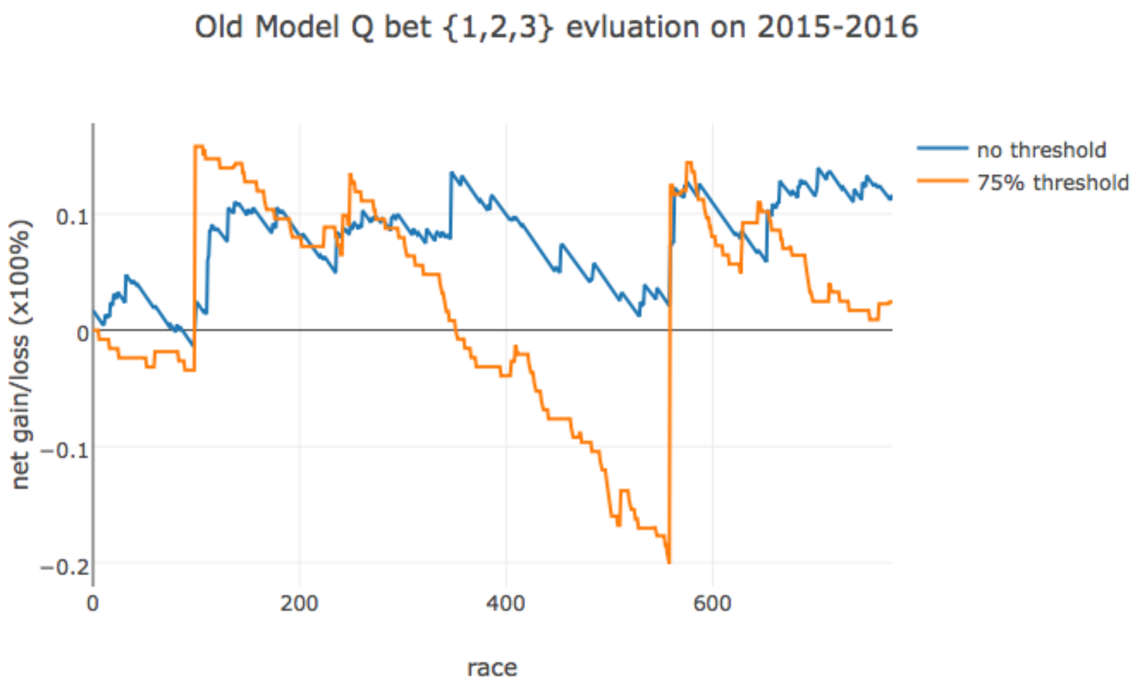


Figure 55

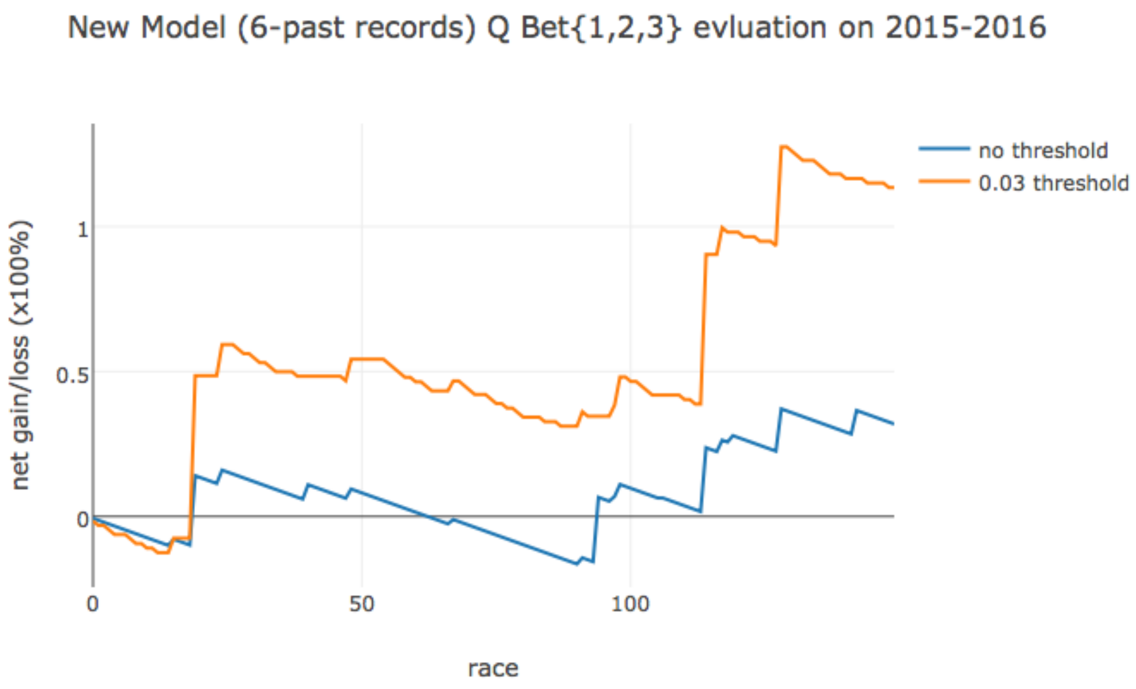

New Model (6-past records) Q Bet{1,2,3} evluation on 2015-2016
no threshold
0.03 threshold
net gain/loss (x100%)
1
0.5
0
0
50
100
race

Figure 56

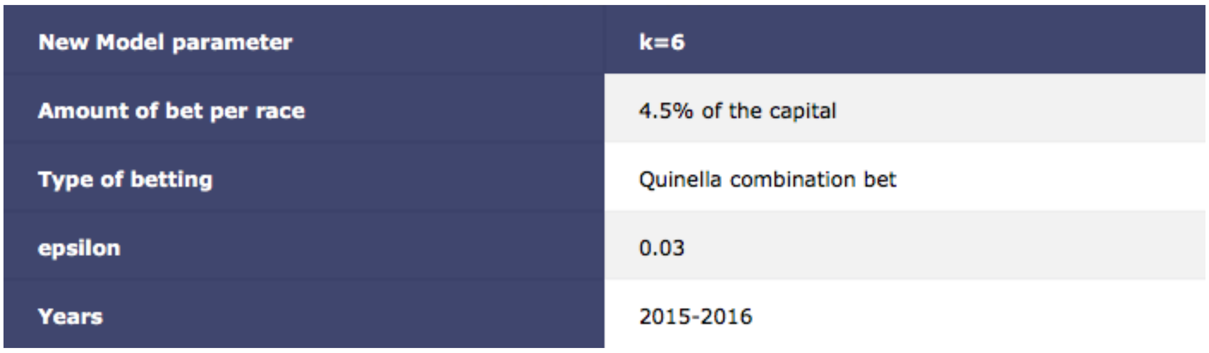
Model Application

The new model is not ready to be used in the real world, it is because from an investor perspective, we should first assume we have finite capital and we should know how much money we are going to bet in each race. However, in the evaluation method we have been using so far, we have assumed we had infinite capital and we were betting 10 dollars per race, we were calculating the gain by the amount of money we won minus the amount of money we spent on the betting after a year. Naturally, we would like to distribute the investor's capital equally on each race, for example if we have \$1000 and we are going to bet on 10 race, the amount of betting per race will be \$100. However, the problem is we don't know how many race we are going to bet, recall that we will only bet on the race with complete information and with threshold 0.03, the problem became an optimization problem and we are not going to solve it in this project.

Naive strategy

The naive way to utilize the models is to come up with a static strategy to decision the amount of betting per race, we arbitrarily created one strategy (Table 12), and tried to apply it to bet on quinella combination bet $\{1,2,3\}$ (Figure 57).

Table 12



New Model parameter	k=6
Amount of bet per race	4.5% of the capital
Type of betting	Quinella combination bet
epsilon	0.03
Years	2015-2016

Result

We were betting 1.5% of the current capital in each race, and we have set the threshold to 0.03. After one year, we were managed to generate over 300% net profit.

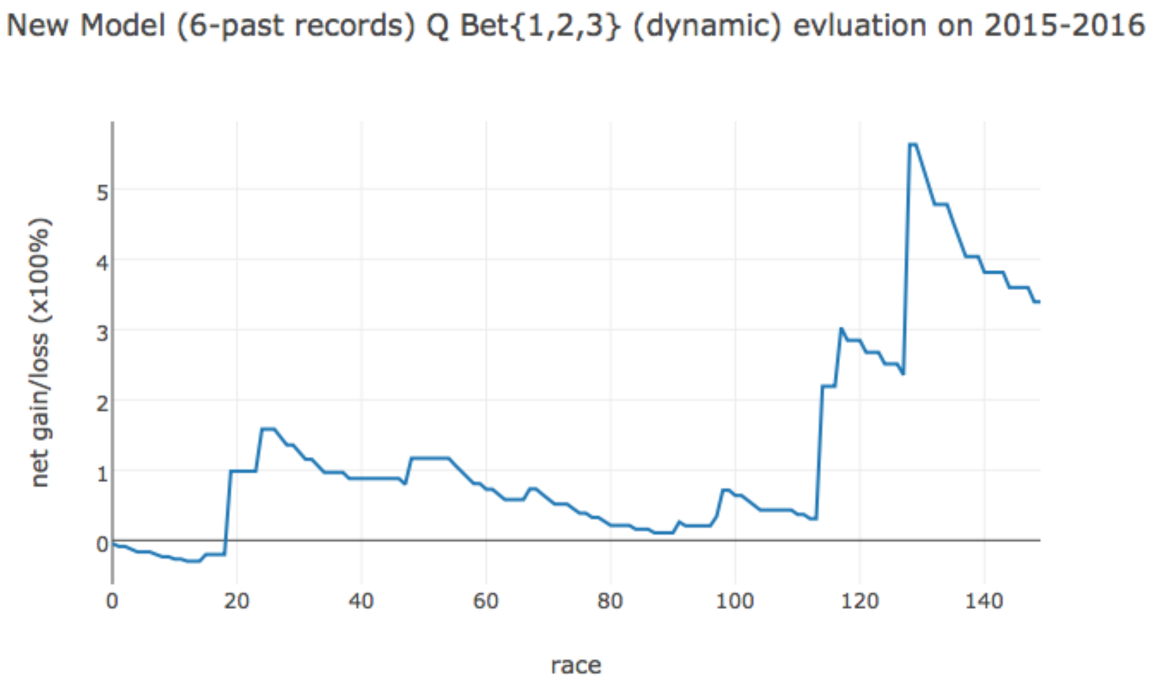


Figure 57

Limitation and difficulties

Web page technology has changed a lot over years so that when we are developing the scraper to collect the data from HKJC, we need to write a lot of exceptional cases, different versions for different years of data.

Also preparing data is time-consuming, in this semester, almost 90% of the time spent on the project were using for structuring the data, formatting the data and filtering the data.

Moreover, although HKJC provided lots of data on their website, some important data have not disclosed, such as opening odds, official horse's age, jockey’s health etc. A much more accurate model could be developed if more data could be reached.

Conclusion

To conclude, we have trained a linear regression model, which is predicting the finishing time of a horse, such that it performed well on different kinds of betting. We also show that the higher the k-value, which means the more features we could capture for a model, the better the model could perform. We also discussed under a special condition, the model will perform even better. However, the model we trained this semester doesn’t perform well on the win bet.

Acknowledgements

We would like to express our sincere thanks and appreciation to Professor LYU Rung Tsong Michael for giving clear guidance, effective opinions and encouragement. Also, we would like to thanks Mr. Edward Yau for brainstorming and having deep discussions with us.

Bibliography

[1] N. M. A. a. D. Merritt, "SUCCESSFUL PREDICTION OF HORSE RACING RESULTS USING A NEURAL NETWORK," Merritt, N M Allinson and D, UK, 2015.

[2] R. S. L. R. Sameerchand Pudaruth, "Generating Horse Racing Tips at the Champs De March Using Fuzzy Logic," IJCST Vol. 4, Mauritius, 2013.

[3] A. DAVIS, "Profitable Strategies in Horse Race Betting Markets," UNIVERSITY OF MELIBOURNE DEPARTMENT OF MATHEMATICS AND STATISTICS, MELBOURNE, 2013.

[4] Z. I. I. Gavazang, "Horse Racing Prediction using Artifical Neural Network," in *NNECFS*, Zanjan, 2010.

[5] T. E. o. E. Britannica, "Britannica," 10 February 2017. [Online]. Available: https://global.britannica.com/sports/horse-racing. [Accessed 10 March 2017].

[6] P. Lynch, "What Is A Neural Network & How Is It Used In Horse Racing?," RACEADVISOR, 16 December 2015. [Online]. Available: http://www.raceadvisor.co.uk/neural-network-used-horse-racing/. [Accessed 1 Februray 2017].

[7] S. D. a. B. Townson, "Predicting Horse Racing Outcomes in India," NYC DATA SCIENCE ACADEMY, 22 September 2016. [Online]. Available: https://blog.nycdatascience.com/student-works/horse-racing/. [Accessed 2 January 2017].

[8] ScienceDailty, "Web crawler," ScienceDailty, 10 December 2016. [Online]. Available: https://www.sciencedaily.com/terms/web_crawler.htm. [Accessed 20 December 2016].

[9] Wikipedia, "Wikipedia - Hong Kong Jockey Club," Wikipedia, 2 December 2016. [Online]. Available: https://en.wikipedia.org/wiki/Hong_Kong_Jockey_Club. [Accessed 3 December 2016].

[10] sradack, "Generalizing the ELO Rating System for Multiple Players," Blogspot, 3 12 2016. [Online]. Available: http://sradack.blogspot.hk/2008/06/elo-rating-system-multiple-players.html.

[11] H. K. J. Club, "Special Incidents Index," Hong Kong Jockey Club, 3 December 2016. [Online]. Available: http://www.hkjc.com/English/include/special_race_index.htm. [Accessed 10 January 2017].

[12] racealyst, "NEURAL NETWORK DIARY #7: CONVERTING DATA TO USABLE FORM," RACEALYST, 12 September 2015. [Online]. Available: http://racealyst.com/tag/artificial-neural-networks/. [Accessed 3 September 2016].

[13] J. Brownlee, "Linear Regression for Machine Learning," machinelearningmastery, 25 March 2016. [Online]. Available: http://machinelearningmastery.com/linear-regression-for-machine-learning/. [Accessed 2 January 2017].

[14] N. Matlof, "From Linear Models to Machine Learning," The Chapman & Hall imprint, California, 2016.

[15] L. F. G. E. Brandon Rohrer, "How to choose algorithms for Microsoft Azure Machine Learning," Microsoft, 14 March 2017. [Online]. Available:

https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice. [Accessed 1 April 2017].

[16] D. Mesquita, "Big Picture Machine Learning: Classifying Text with Neural Networks and TensorFlow," Medium, 9 April 2016. [Online]. Available: https://medium.freecodecamp.com/big-picture-machine-learning-classifying-text-with-neural-networks-and-tensorflow-d94036ac2274. [Accessed 10 April 2017].

[17] N. HARVEY, "Intro to Machine Learning using Tensorflow – Part 1," 25 January 2017. [Online]. Available: https://blog.openshift.com/intro-machine-learning-using-tensorflow-part-1/. [Accessed 2 February 2017].

[18] pythonprogramming, "Introduction to Deep Learning with TensorFlow," pythonprogramming, 19 July 2016. [Online]. Available: https://pythonprogramming.net/tensorflow-introduction-machine-learning-tutorial/. [Accessed 2 January 2017].

[19] K. HUGHES, "TensorKart: self-driving MarioKart with TensorFlow," KEVIN HUGHES , 28 December 2016. [Online]. Available: http://kevinhughes.ca/blog/tensor-kart. [Accessed 3 January 2017].

[20] I. Polosukhin, "TensorFlow Tutorial— Part 1," Meduim, 19 Novmeber 2015. [Online]. Available: https://medium.com/@ilblackdragon/tensorflow-tutorial-part-1-c559c63c0cb1. [Accessed 20 July 2016].

[21] I. Polosukhin, "TensorFlow — Text Classification," Meduim, 20 Novmeber 2016. [Online]. Available: https://medium.com/@ilblackdragon/tensorflow-text-classification-615198df9231. [Accessed 40 December 2016].

[22] I. PolosukhinFollow, "TensorFlow: Combining Categorical and Continuous Variables," Meduim, 28 October 2016. [Online]. Available: https://medium.com/@ilblackdragon/tensorflow-tutorial-part-4-958c29c717a0. [Accessed 30 December 2016].

[23] I. Polosukhin, "Tensorflow Tutorial — Part 2," Meduim, 24 Novemebr 2015. [Online]. Available: https://medium.com/@ilblackdragon/tensorflow-tutorial-part-2-9ffe47049c92. [Accessed 30 December 2016].

[24] I. Polosukhin, "TensorFlow Tutorial — Part 3," Meduim, 1 February 2016. [Online]. Available: https://medium.com/@ilblackdragon/tensorflow-tutorial-part-3-c5fc0662bc08. [Accessed 30 December 2016].

[25] S. RAY, "Tutorial – Getting Started with GraphLab For Machine Learning in Python," analyticsvidhya, 3 December 2015. [Online]. Available: https://www.analyticsvidhya.com/blog/2015/12/started-graphlab-python/. [Accessed 30 December 2016].

[26] roos, "Building Interactive Dashboards with Jupyter," dominodatalab, 11 November 2015. [Online]. Available: https://blog.dominodatalab.com/interactive-dashboards-in-jupyter/. [Accessed 30 December 2016].

[27] S. Mead, " PostgreSQL Schema Visualization," openscg, 22 December 2016. [Online]. Available: https://www.openscg.com/2016/12/postgresql-schema-visualization/. [Accessed 2 February 2017].

[28] oldschoolvalue, "Apply the Kelly Criterion to Investing and Your Portfolio Sizing," oldschoolvalue, 18 June 2014. [Online]. Available:

https://www.oldschoolvalue.com/blog/investing-strategy/kelly-criterion-investing-portfolio-sizing/. [Accessed 30 December 2016].

[29] D. H. plus, "The Importance of Staking Plans – Kelly Criterion," sbo, 1 January 2016. [Online]. Available: https://www.sbo.net/strategy/importance-staking-plans-kelly-criterion/. [Accessed 30 December 2016].

[30] S. RAY, "Beginner’s guide to Web Scraping in Python (using BeautifulSoup)," analyticsvidhya, 22 October 2015. [Online]. Available: https://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-python/. [Accessed 30 December 2016].