The Chinese University of Hong Kong

FINAL YEAR PROJECT REPORT (TERM 1)

Applying Reinforcement Learning to “Play” Horse Racing

Author:

CHEUNG Kam Fung, OR Ka Yui

Supervisor:

Prof. Michael R. LYU

LYU2003

Faculty of Engineering

Department of Computer Science and Engineering

December 2, 2020

THE CHINESE UNIVERSITY OF HONG KONG

Abstract

Faculty of Engineering

Department of Computer Science and Engineering

BSc degree in Computer Science

Reinforcement Learning in Horse Racing

by CHEUNG Kam Fung, OR Ka Yui

Reinforcement Learning has become popular since the appearance of Alphago. It has been applied to multiple areas in the world and now we would like to study and apply reinforcement learning in horse racing, which is the most famous gambling event in Hong Kong. In this report, we collect data from Hong Kong Jockey Club and Hong Kong Observatory. After analysing them, we first use XGBoost regressor to build a model to predict the horse racing result. Then, we will construct the Deep Q Network model using the result of XGBoost and conclude our work. This is called a model based Reinforcement Learning.

Acknowledgements

We truly appreciate our supervisor Professor Micheal R. Lyu and advisor Mr. Edward Yau for guiding and supporting this project.

Also we would like to express our special thanks to the previous project studying reinforcement learning and horse racing. They give us a good reference to our project “Reinforcement learning in horse racing”.

Content Page

Chapter 1 Introduction

1.1 Overview

The topic of this final year project is trying to learn how to bet on horse racing with rein, throughout this report we will demonstrate the work done during the first semester. This chapter offers a brief overview of this final year project and introduction to the topic. Moreover, it provides related work and previous approaches on the horse racing predictions. In the end, it introduces the difficulties in predicting horse racing results and the coming objective in the second semester.

1.2 Motivation

Nowadays, artificial intelligence has become the most popular technique in the world. As one of the areas of artificial intelligence, machine learning is applied to many different uses in the world, and many softwares and hardwares have born because of machine learning. For example, many CPUs and GPUs have claimed that they are more powerful in machine learning. The most popular electric car company in the world, Tesla, has also developed the Autopilot AI using deep neural networks [1]. Google created the best AI chess player Alphago which beat all human chess players [2]. The trend of machine learning attracts us to study about machine learning and apply machine learning to one of the most popular events in Hong Kong, horse racing. The gambling of horse racing can bring lots of money to us. This attracts us to study horse racing.

Before this project, there are the other projects to apple machine learning in horse racing. For example, LYU1603 Final Year Project used Tensorflow to predict the horse racing result [3]. LYU1703 Final Year Project used a deep neural network to predict the result [4]. LYU1805 Final Year Project used deep probabilistic programming [5]. All of them used supervised learning to predict the result. Instead of supervised learning, we are going to use reinforcement learning.

Reinforcement learning becomes famous in these years due to Alphago who has used reinforcement learning. Before that, people have used reinforcement learning to play different games. However, no one has ever used reinforcement learning to predict the result of horse racing or teach the AI to bet. That is why we would like to apply reinforcement learning in horse racing.

1.3 Backgound

1.3.1 Horse Racing

Horse racing has been popular for many years in Hong Kong. This event started in the 1840s when Hong Kong was not yet transferred to China [6]. With the complete and strict law of gambling, Hong Kong citizens love betting in horse racing. In 2019/2020, Hong Kong Jockey Club has recorded the total amount bet of horse racing by customers is around 121.0 billion [7].

There are multiple components of horse racing including racecourse, system, horse, jockey, trainer and wager [8].

1.3.1.1 Basic of Horse Racing

There are two main racecourses in Hong Kong. They are Sha Tin Racecourse and Happy Valley.

	Mainly Host	Number of races per race meeting	Track	Max. Number of starters
Sha Tin Racecourse	Sunday day races	10	Turf of All-Weather	14
Happy Valley Racecourse	Wednesday night races	8	Turf	12

The system of horse racing includes season, distance, class, rating, weight and draw. The race season is from September to July with 99 race meetings. Distance can be classified as short, middle and long.

Short	1000M	1200M	1400M
Middle	1600M	1650M	1800M
Long	2000M	2200M	2400M

The horses are classified from Class 1 to Class 5 where Class 1 is the highest class and Class 5 is the lowest. A new horse starts with 52 rating. Around 5 to 7 points will be added to the rating for each win. First 4 in the race will add points while losing in the race will deduct points.

Rating	Class
>=85	1
>=80 & <=100	2
>=60 & <=80	3
>=40 & <=60	4
>=0 & <=60	5

There are handicaps which means runners carry different weights to equalise their chances of winning. The higher rating horse carries more weight. Also, with more additional weight, the horse will get more rating if it wins.

The draw refers to the horse's starting position in the starting gate. The smaller number of the draw means the closer to the inside rail. The horse may get a different advantage from the draw depending on the horse's characteristics.

The jockey rides the horses according to the trainers' instruction. The trainer trains and cares for the horses and manages operation. They all get licenses from Hong Kong Jockey Club.

1.3.1.2 Bet

Odds decides how much you get if you win. Usually, higher odds mean lower chance of winning. For example, if the win odds is 1.5 and you win with $10, you will get the dividend $15 back including the cost $10. There are two types of odds which are Win odds and Place odds. Now, the betting type of horse racing:

Simple bet:

Type	Condition of winning
Win	Pick the winner
Place	Pick any one of the first three horses
Quinella	Pick the first and second horses in any order
Quinella Place	Pick any two of the first three horses in any order
Banker	Pick a banker which must be in all possible combinations and three other selections to pair with the banker. The Quinella combination gives a dividend.
Multiple	Pick 4 horses and there will be 6 combinations. Each Quinella Place combination gives a dividend.
3 Pick 1	The horses are in 3 groups as {1st,2nd}, {3rd,4th,5th}, others. The dividend is given by any runner from the chosen group wins. The odds are special.

Exotic Bets (Single-race):

Type	Condition of winning
Forecast	Pick the first and second horses in correct order
Trio	Pick the first three horses in any order
Tierce	Pick the first three horses in correct order
First 4	Pick the first four horses in any order
Qurtet	Pick the first four horses in correct order

Exotic Bets (Multiple-race):

Type	Condition of winning
Double Trio	Pick the first three horses in any order in each of the two nominated races
Triple Trio	Pick the first three horses in any order in each of the three nominated races
Double	Pick the winner in each of the two nominated races
Treble	Pick the winner in each of the three nominated races (Only available in the last 3 races of each race meeting)
Six Up	Pick the winner in each of the six nominated races (Only available in the last 6 races of each race meeting)
All Up	Placing bets on more than 1 race. They can only be simple bets' combinations. It can be choosing n races with m combinations. The odds will be multiplied by each other.

1.3.2 Reinforcement Learning

Reinforcement learning is one kind of machine learning. It is about how the agent learns to take actions in the environment to get the maximum reward [9]. Reinforcement learning is totally different from the other two kinds of machine learning. Supervised learning trains the agent to learn from the data with labeled input and output. Unsupervised learning trains the agent to learn from the data with only input and find the pattern. More details of reinforcement learning will be mentioned in the chapter of reinforcement learning.

1.4 Objective

In this project, our objective is to apply reinforcement learning into horse racing. We want to build a model that can place the bet on the winning horse by predicting it. Before the model can have positive profit, we would like the model to gamble like a

human being. We will focus on the strategy that the model takes. For example, the model chooses to place a bet on which horses or chooses not to bet.

After training the model similar to human gambling, we would like to push the limit to go beyond human to gain profit in the long term or short term. We want the model to be overwhelming in horse racing. The result will be compared to the other method of machine learning in horse racing.

Chapter 2 Data Preparation

2.1 Data Collection

There are companies selling historical data from 2000 to 2020. But due to our limited budget, we collect our dataset from the Hong Kong Jockey Club official website [10] and Hong Kong Observatory official website [11].

2.1.1 Information of Horses

The information such as weight, sex, color, dim, etc. are collected. The dataset contains these features from 2771 horses.

2.1.2 Historical Horse Racing Records

All the horse racing records from 2010 to 2020 are collected. Each horse participates in horse racing for an average of 3 to 4 years. Therefore, the horse racing records between 2014 to 2019 will be mainly used in our model as we want to label the horse in our model. The dataset contains 37,755 records and 3,080 races from 2014 to 2019. And 2019 to 2020 will be our testing set with 810 races and 9859 records.

2.1.3 Historical Weather Data

The average degree, pressure and humidity from 2010 to 2020 are collected. The data between 2014 to 2019 will be mainly used for our xgboost regressor and deep neural network regressor, as we only use the horse racing record from 2014 to 2020 to train our model. The dataset contains 37,755 records for each race from 2014 to 2019. And 2019 to 2020 will be our testing set with 9859 records.

2.2 Data Description

2.2.1 Race data

The following tables show the feature values of racing records obtained from the HKJC official website. There are around 3890 race records from 2014 to 2020. (End of the 2019.)

Features	Description	Type	Values
race_date	The date of the race	Index	/
race_no	The numer of a race in a day	Index	/
race_index	Unique id of the race	Index	/
location	Location of the race	Categorical	HV, ST
class	Class of the horses	Categorical	Class 1 to 5, Group 1 to 3
race_length	Distance of the race	Categorical	1000, 1200, 1400, 1600, 1650, 1800, 2000, 2200, 2400
course	Track of the race	Categorical	A, A+3, B, B+2, C, C+3
draw	Draw of the horse in a race	Categorical	14 distinct values
going	Condition of the track	Categorical	FAST, SLOW, WET FAST, WET SLOW, FIRM, GOOD TO FIRM, GOOD, GOOD TO YIELDING, YIELDING,
horse_id	Unique id of the horse	Categorical	2744 distinct values
jockey_name	Unique id of jockey	Categorical	113 distinct values
trainer_name	Unique id of trainer	Categorical	112 distinct values
actual_weight	Weight added to the horse	Real value	/
declared_horse_weight	Weight of the horse	Real value	/
win_odds	The odds of betting the horse	Real value	/
place	The final place of the horse in a race	Categorical	14 distinct values
finish_time_sec	Finishing time of the horse in a race	Real value	(Seconds)

Figure 1. Race data

2.2.2 Horse data

The below tables recorded the data of the horse. This record includes all horse data in each race. This means the same horse has different data in different races

because these data will change in each race. For example, after the horse may increase its rating, weight from the previous race.

Features	Description	Type	Values
last_actual_weight	The actual weight of last race	Real Value	/
last_declared_horse_weight	The last weight in last race	Real Value	/
diff_actual_weight	Difference actual weight between present race and last race	Real Value	/
diff_declared_horse_weight	Difference declared weight between present race and last race	Real Value	/
country	the country of the horse	Categorical	US,AUS,etc.
age	The age of the horse	Real Value	/
colour	The colour of the horse	Categorical	Bay, Chestnut, etc.
sex	The sex of the horse	Categorical	Gelding
import_type	The import type of the horse	Categorical	PP,PPG
sire_name	The name of the horse's sire	Categorical	Acclamation, Patagan,etc.
last_plc	The place in the last race	Index	/
last_rating	The rating in the last race	Index	/
rating	The rating now	Index	/

Figure 2. Horse data

2.2.3 Weather data

The below table is the weather data in each race. There is 3890 weather data which is every day of the race. Compared to the previous data (LYU1703), the feature values of the weather are significantly fewer. It is because we believe that some weather data will not help but only make noise to the model. For example, the moon phase of the race day is supposed not to affect the result of the horse because the race court gets a lot of lighting.

Features	Description	Type	Values
mean_degree	The mean of the temperature of the race day	Real Value	/
mean_humidity	The mean of the humidity of the race day	Real Value	/
mean_pressure	The mean of the air pressure of the race dat	Real Value	/

Figure 3. Weather data

2.2.4 Additional data

Features	Description	Type	Values
total_first_count	The total count of first place	Real Value	/
total_second_count	The total count of second place	Real Value	/
total_third_count	The total count of the third place	Real Value	/
total_race_count	The total count of joined race	Real Value	/

Figure 4. Additional data

We believe that if a horse wins a lot, then it is likely to win the next race. Therefore, we add these features to increase the accuracy of our prediction. We have also analyzed the correlation between these features in the next sub-chapter.

2.3 Data Analysis

In this part, we are going to find the relationship between different features values based on the previous project analysing the data. We will classify the data into continuous data and categorical data [12].

2.3.1 Continuous data

The below table shows the correlation between two continuous values. Here, we will pick some significant values and analyze them.

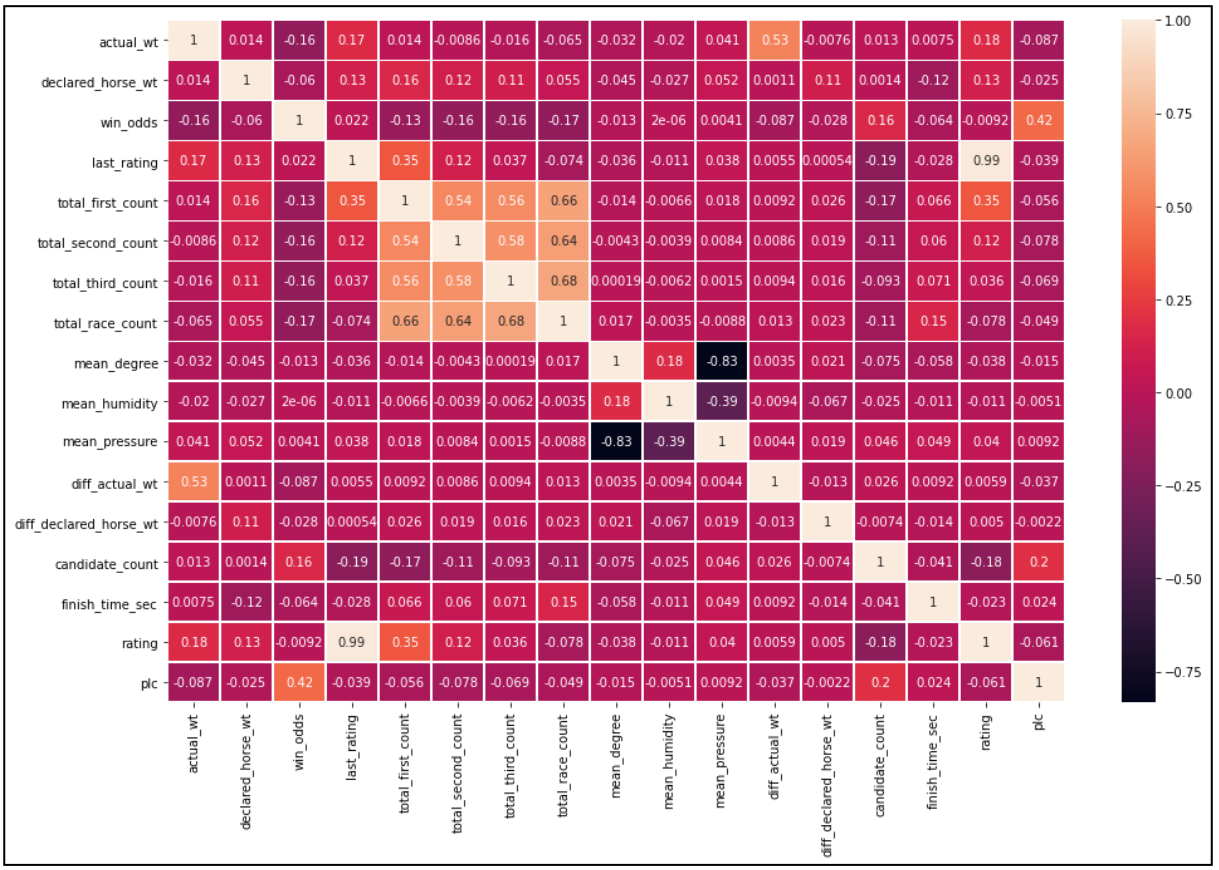

Figure 5. Corr. matrix of Continuous data

The correlation between the count of the first place and second place is 0.54, which is pretty high, and it implies that if a horse wins a lot before, then it is likely to win more. The correlation between the count of the first place and third place and the count of the second place and third place are similar.

The correlation between win odds and the place is 0.42. Supposing that the lower win odds get higher place and this correlation has proved that. The place is the index which is 1 is the highest and the correlation makes sense.

The rating is also related to the count of first place, A horse wins more, then the rating will increase. And there is an interesting correlation that the actual weight of a horse is positively correlated to the horse's rating with 0.18.

2.3.2 Categorical data

The below table shows the association between two categorical values. Here, we will pick some significant, meaningful values and analyze them. The association means the probability of getting the other data if we have data.(Forecast) This is called “Theil’s U” or “Uncertainty coefficient”. For example, if we have the horse_id, we can forecast the country, colour, sex, import type and sire name. This is certain because a horse can only have one country, one colour, one sex, one import type and one sire name. However, we can still find some interesting associations.

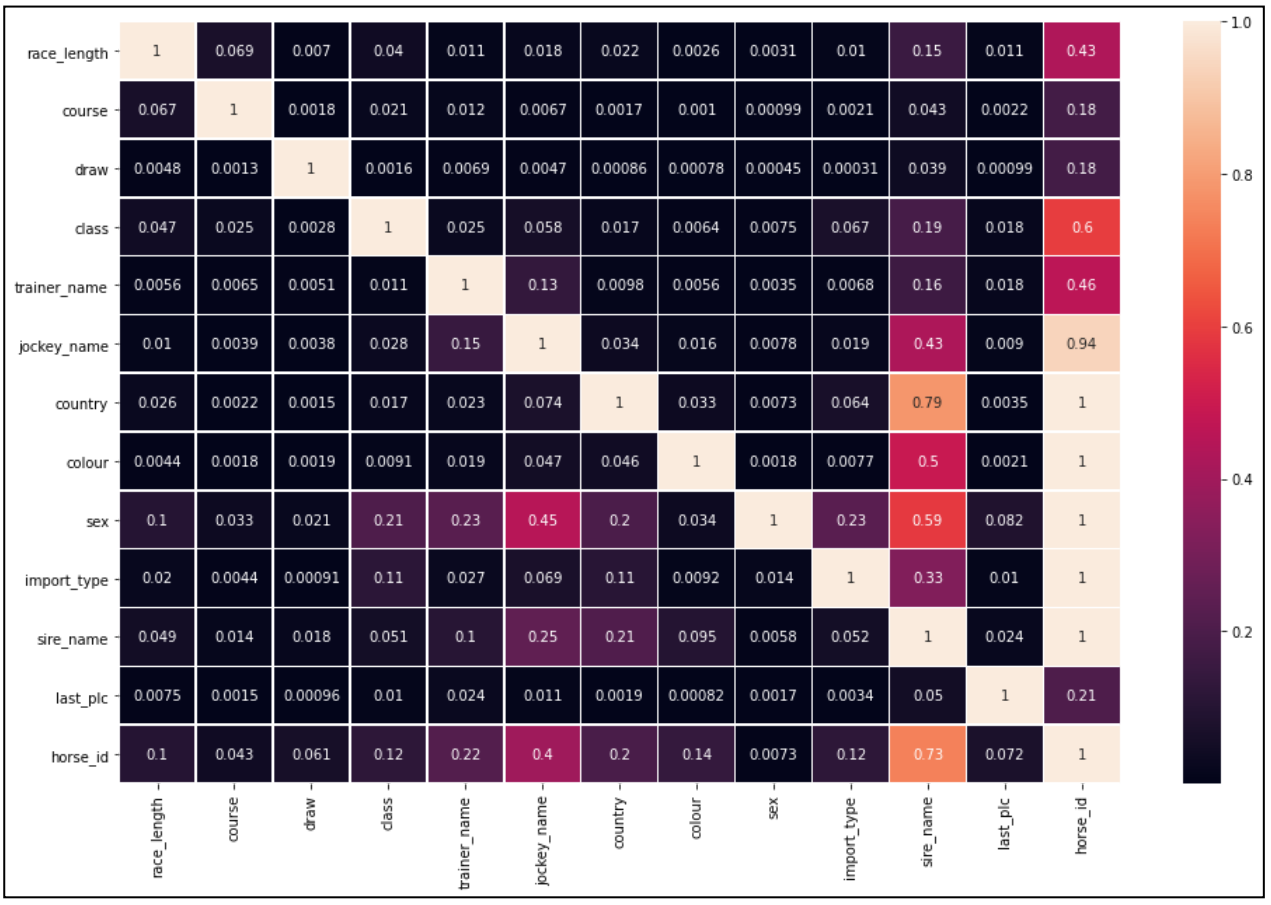

Figure 6. Theil’s U of Continuous data

The association between the jockey name and horses’ id is 0.94. This means most of the time the horse will only be rode by one jockey. The other association is the class and the race length. These mean the horse will usually train for a specific race length instead of changing it at a high frequency. The horse will stay in one class instead of changing in.

2.3.3 Continuous data vs categorical data

The below table shows the correlation ratio between numerical values and categorical values.The correlation ratio will tell the probability of knowing the category if we have a number. Here, we will pick some significant values and analyze them.

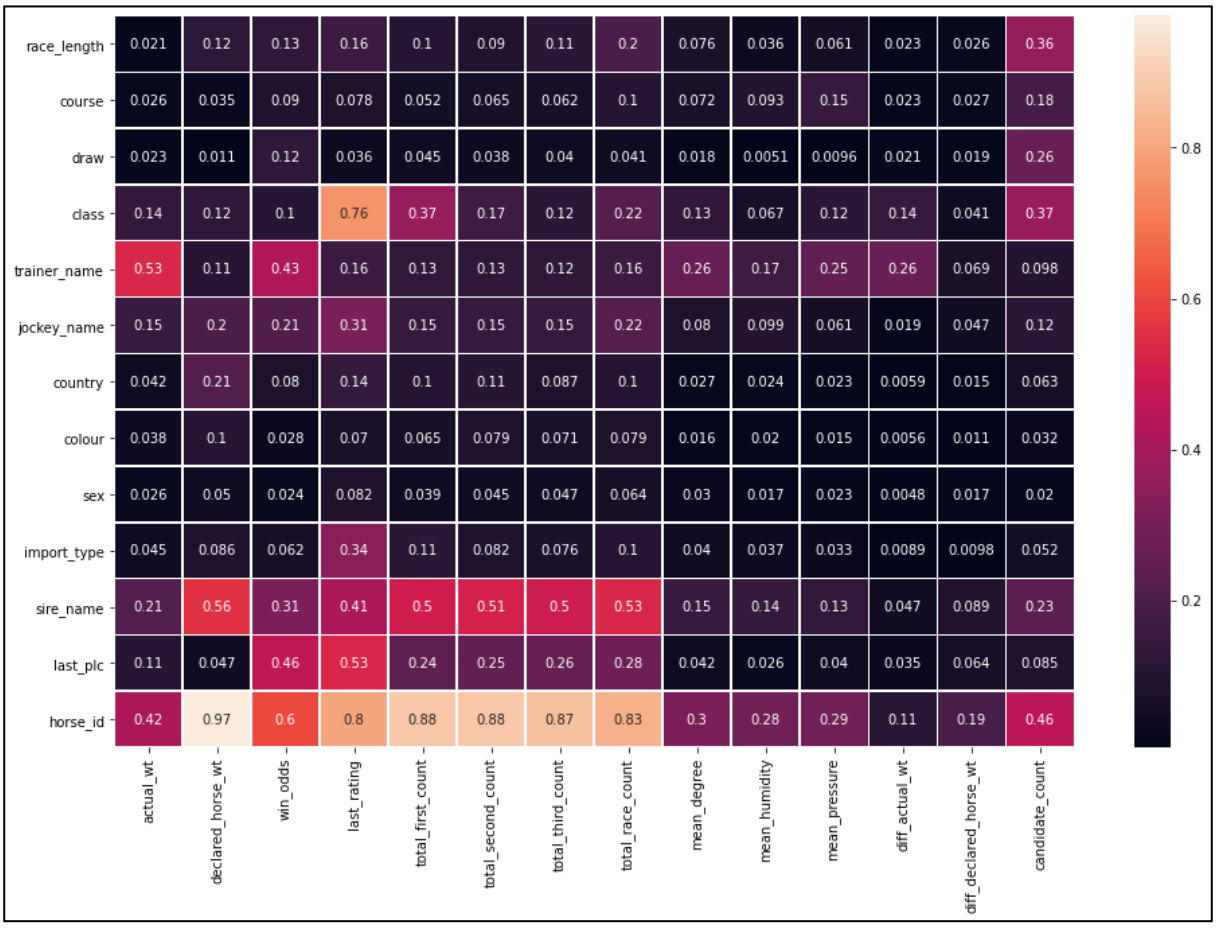

Figure 7. Correlation ratio between Continuous data and Categorical data

For example, if we have the last rating of the horse, in 76% we will know what class the horse belongs to. In this graph, we shall see if we have numerical data, we will easily know which horse it is compared to the other data. This proves that most of the horses have unique data instead of the same data but different horses.

2.3.4 Conclusion of analysis

In machine learning, if there are highly correlated features, It harms the performance of the model which is called multicollinearity. So we may need to exclude those

features with high correlation. Luckily, there are no features with more than 0.8 correlation between them, So we will use all of these feature values.

2.4 Data preprocess

We normalize all of the continuous data using Z-score normalization other than max-min normalization as we think that there are outliers between the horses. We want a normalization to handle those outliers. We will use one hot encoding on the categorical data.

Chapter 3 XGBoost

3.1 Description

XGBoost was developed by Tianqi Chen in 2014 as a scalable end-to-end tree boosting system [13]. The reason for using XGBoost in this project is there are a lot of feature values like horses' data. Multivariable regression can be effectively done by XGBoost. In this chapter, the principle of XGBoost will be simply introduced and mainly introduced how to apply XGBoost into our model.

3.1.1 Decision Tree

Decision Tree is the basis of the tree boosting system. Here is a decision tree from the paper "XGBoost: A Scalable Tree Boosting System" written by Tianqi Chen. Decision tree classifies the data by asking the input some questions. The tree below is classifying the input by asking whether the input's age and gender. Every node (split) in the tree represents the features of input. The leaf of the tree is the category. There is a prediction score below the category. Decision trees can handle some missing feature in some input which benefits a lot in our case like setting missing in yes or no.

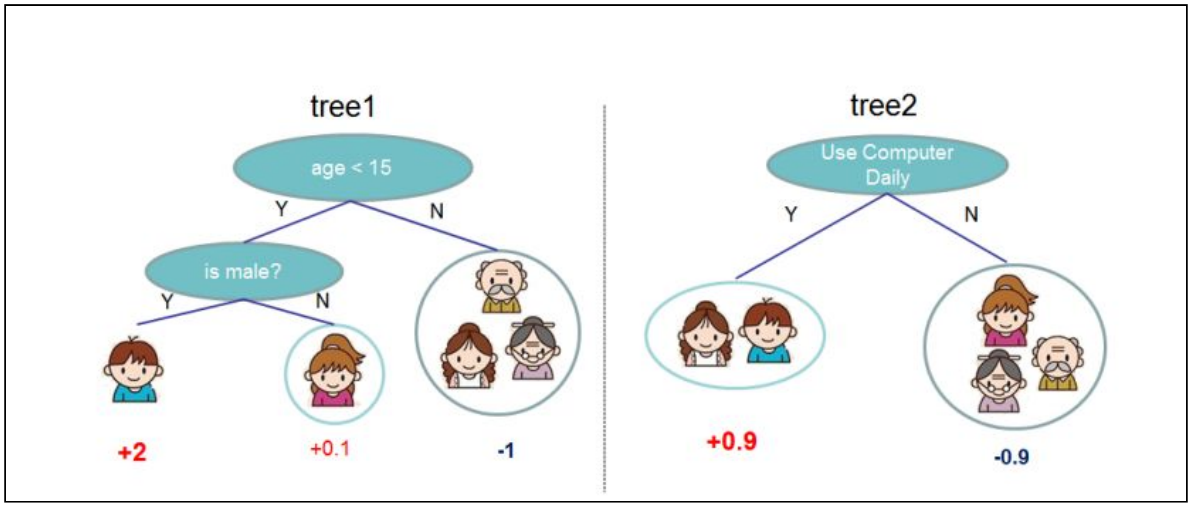

Figure 8. Example of decision tree[1]

[1]A figure in the paper "XGBoost"
URL:https://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf

In our case, the leaf is the predicted weight of every feature value and the predicted weight of each feature value will be used to calculate the predicted finish time. We will use the predicted finish time to predict the finished position of each horse.

However, decision trees usually cause overfitting problems which means training data's accuracy will be very high, while the testing data cannot fit the model, especially there are a lot of feature values causing a lot of noise.

3.1.2 Gradient Boosting Decision Tree

As mentioned in the last part, just the decision tree is difficult to provide a good model to fit the testing data. Boosting is combining weak learners to become a strong learner [14]. Weak learners mean which prediction is not accurate. Strong learners mean their prediction is accurate.

Tree boosting means there are a lot of decision trees with low accuracy, and then combined to a boosted tree with higher accuracy [16]. This is also called a tree ensemble. There is a boy in the figure above. If two trees combine together, the function of boy will become: f(boy) = (+2) + (+0.9) = 2.9

If there are many trees, gradient boosting will be used. The prediction of the i-th tree [15]:

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i)$$

Formula 1. The prediction of the i-th tree

Therefore, the formula of the t-th $\hat{y}_i^t = \hat{y}_i^{t-1} + f_t(x_i)$. $f_t(x_i)$ is the prediction where is the upcoming t-th tree and $\hat{y}_i^{t-1}$ is the prediction which is combined [15]. These two formulas will be used in XGBoost.

From the formula, the boosting will be done step by step. That is why it is called the gradient boosting decision tree.

3.1.3 XGBoost (Extreme Gradient Boosting)

XGBoost is based on gradient boosting decision trees. Here, the derivation of the learning objective is done by Corey Wade and Kevin Glynn from “Hands-On Gradient Boosting with XGBoost and scikit-learn” since this derivation is better for understanding.

The learning objective for t-th boosted tree:

$$obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_l^t\right) + \sum_{i=1}^{t} \Omega\left(f_t\right)$$

Formula 2. The learning objective for t-th boosted tree

The first part of the function is called the loss function which is the mean squared error for regression. is the target value in the i-th row and $\hat{y}_l$ is the prediction in the i-th row. The sum of the difference between all rows will be the error.
Here, the second part is called regularization term which smooths the final learnt weights to avoid overfitting by penalizing the complexity of the model [13].

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

Formula 3. Regulariztion term

W gives the corresponding leaves and T is the number of trees. γ and λ are the penalty. If the penalty is zero, the learning objective will be the same as gradient boosting decision tree’s objective.

Then, Chen used second-order approximation to optimize the learning objective which is from Taylor’s formula. He calculated the first order and second order gradient by $g_i = \partial_{\hat{y}_l t-1} l\left(y_i, \hat{y}_i^{t-1}\right)$and $h_i = \partial_{\hat{y}_l}^2 t_{-1} l\left(y_i, \hat{y}_i^{t-1}\right)$, we can get:

$$obj^{(t)} = \sum_{i=1}^{n} g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$

Formula 4. The learning objective for t-th boosted tree

Let $G_j = \sum_{i \in I_j} g_i$ and $H_j = \sum_{i \in I_j} h_i$ and the optimal weight of a leaf j:

$$w_j = -\frac{G_j}{H_j + \lambda}$$

Formula 5. The optimal weight

The final objective function:

$$obj^{(t)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

Formula 6. Final objective function

This function can measure the quality of a tree by how well the model fits the data [13] The smaller score means the better structure.

In boosting, everytime the best score is calculated by best weight. However, it is impossible to find all possible tree structures. Therefore, exact greedy algorithm is used to find the best tree [13]:

$$\mathcal{L}_{split} = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma$$

Formula 7. Exact greedy algorithm

From depth = 0, starting from a single leaf and adding branches to the tree iteratively. [13] γ and λ are the penalty.

The first term in the Eqn is the score of the left child tree. The second term is the score of the right child tree. The third term is the score of not being a branch of the tree. Therefore, a loop is required to find out the best tree.

To conclude, XGBoost would find out the best G and H in the final objective function by ensembling trees. Then using an exact greedy algorithm to find the best branch adding to the tree. Here is the full algorithm [13] :

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted*$(I,$ *by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

Algo 1. Exact Greedy Algorithm

XGBoost uses shrinkage to avoid overfitting. Shrinkage adds weight after each step of tree boosting [13]. The concept of shrinkage is similar to learning rate.

3.2 Benefit

There are some benefits of choosing XGBoost. Block compression and sharding improves the speed of reading data from the disk. Cache-aware access provides faster access to cache for calculation.

The column block for parallel learning can reduce the time of sorting the data.

The data will be put into blocks and parallel computing can be used to reduce the time consumed. Multiple cores of the cpu can be used.Since this project's data is very large, handling the data and combining multiple trees at the same time can effectively reduce time. Therefore, using XGBoost can improve the performance of regression.

Here is the data provided by Chen in “XGBoost: A Scalable Tree Boosting System”:

Method	Time per Tree (sec)	Test AUC
XGBoost	0.6841	0.8304
XGBoost (colsample=0.5)	0.6401	0.8245
scikit-learn	28.51	0.8302
R.gbm	1.032	0.6224

XGBoost’s performance is much better than other exact greedy methods like scikit-learn and R.gbm.

3.3 XGBoost in Horse Racing

3.3.1 Progress

First, the target is to predict the horse’s finishing time and find out the ranking of the horses and gamble by the result. Therefore, we will use XGBoost to find out which feature value affects the finishing time the most. We assume that we do not know which feature value so we will put all of them into the algorithm.

3.3.1.1 Data

The training data will be the data from 2014 to 2018. The testing data will be the data from 2019.

The configuration will be the race date, season, race index, horse id, race number, class, win odds, total race counted. These are used to group up the data and most of them will not affect the finishing time.

The labeled input is the rating of the horse and the place of the horse in the race. These two data are labeled since we think they affect the finishing time the most. The labeled output will be the finishing time in seconds which is the prediction needed.

For the feature values, there are two kinds of feature values, continuous features value and categorical features. Continuous features mean the features can be

represented in number. In our case, they include actual weight, declared horse weight, win odds, last rating, total count in first place, total count in second place, total count in third place, total count of race taken, mean degree of the day, mean humidity of the day, mean pressure of the day, different between actual weight and mean weight of the race, different between declared weight and mean weight of the race, and the number of candidates in the race.

Categorical features are transferred to true and false. For example, there are multiple race lengths like 1000M, 1200M, 1400M, 2000M, 2200M. They will be transferred to true and false by {0,1} where 0 represents false and 1 represents true. The node of the tree will be (Categorical features < 0.5) to split. The categorical features include the length of the race, the course, the draw, the class, the name of trainer, the name of the jockey, country of the horse, colour of the horse, sex of the horse, the import type, sire name, last place and horse id.

Here is part of the training data and part of the testing data:

Excluding the label, we have 37755 x 3963 training data.

	df_date	race_date	season	race_index	horse_id	race_no	class	win_odds_	total_race_count_	finish_time_sec	rating	plc	actual_wt	declared_horse_wt	win_odds	l
0	2014-01-01	2014/01/01	13/14	286	HK_2008_K364	1	Class 5	13.0	74	83.47	23.0	2	-1.432731	-0.457444	-0.416353	
1	2014-01-01	2014/01/01	13/14	286	HK_2012_P077	1	Class 5	2.9	9	83.89	40.0	3	1.593909	-0.760515	-0.603499	
2	2014-01-01	2014/01/01	13/14	286	HK_2009_L155	1	Class 5	16.0	39	83.97	33.0	4	0.160238	1.297181	-0.360765	
3	2014-01-01	2014/01/01	13/14	286	HK_2010_M188	1	Class 5	12.0	22	84.09	37.0	5	1.116019	0.802696	-0.434882	
4	2014-01-01	2014/01/01	13/14	286	HK_2011_N158	1	Class 5	23.0	10	84.22	33.0	6	0.160238	-0.266030	-0.231059	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
37750	2018-12-29	2018/12/29	18/19	300	HK_2016_A030	10	Class 2	27.0	22	81.93	85.0	4	-0.476950	0.100846	-0.156941	
37751	2018-12-29	2018/12/29	18/19	300	HK_2018_C143	10	Class 2	368.0	2	83.34	83.0	11	-0.476950	-1.127391	6.161578	
37752	2018-12-29	2018/12/29	18/19	300	HK_2018_C054	10	Class 2	24.0	1	81.75	81.0	3	-0.795543	-0.345786	-0.212529	
37753	2018-12-29	2018/12/29	18/19	300	HK_2017_B257	10	Class 2	1.6	7	81.62	89.0	2	0.478831	0.435820	-0.627588	
37754	2018-12-29	2018/12/29	18/19	300	HK_2017_B309	10	Class 2	33.0	1	83.51	82.0	14	-0.636246	1.759763	-0.045765	

37755 rows × 3975 columns

Figure 9. Part of the training data

Excluding the label, we have 9859 x 3963 data testing data.

	df_date	race_date	season	race_index	horse_id	race_no	class	win_odds_	total_race_count_	finish_time_sec	rating	plc	actual_wt	declared_horse_wt	win_odds
0	2019-01-01	2019/01/01	18/19	301	HK_2016_A344	1	Class 4	2.8	10	94.61	54.0	1	1.275316	1.600252	-0.605352
1	2019-01-01	2019/01/01	18/19	301	HK_2015_V400	1	Class 4	7.0	18	95.10	44.0	2	-0.317653	0.627233	-0.527529
2	2019-01-01	2019/01/01	18/19	301	HK_2017_B006	1	Class 4	11.0	14	95.22	41.0	3	-1.114137	-0.250079	-0.453411
3	2019-01-01	2019/01/01	18/19	301	HK_2016_A120	1	Class 4	17.0	20	95.25	53.0	4	1.116019	-1.159293	-0.342235
4	2019-01-01	2019/01/01	18/19	301	HK_2016_A087	1	Class 4	3.5	13	95.44	41.0	5	-0.795543	0.005139	-0.592382
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9854	2019-12-29	2019/12/29	19/20	293	HK_2017_B161	10	Class 3	10.0	17	100.30	72.0	4	0.638128	0.308211	-0.471941
9855	2019-12-29	2019/12/29	19/20	293	HK_2016_A193	10	Class 3	13.0	45	100.82	63.0	10	-0.795543	-0.824320	-0.416353
9856	2019-12-29	2019/12/29	19/20	293	HK_2018_C197	10	Class 3	3.9	12	100.15	64.0	3	-0.636246	0.547478	-0.584970
9857	2019-12-29	2019/12/29	19/20	293	HK_2017_B203	10	Class 3	25.0	25	100.09	70.0	1	0.319535	-1.015733	-0.194000
9858	2019-12-29	2019/12/29	19/20	293	HK_2014_T098	10	Class 3	12.0	69	100.13	61.0	2	-0.795543	1.504545	-0.434882

9859 rows × 3975 columns

Figure 10. Part of the testing data

3.3.1.2 Training

The best component of hyperparameters is found out by function grid_search:

- The objective is ‘reg:squarederror’ which is the loss function of the learning objective.
- The learning_rate which is the shrinkage is set to 0.05.
- The max_depth which the maximum depth of the tree is set to 5. This value can prevent overfitting since higher depth may cause the model to learn relations specific to the training data.
- The n_estimators is set to 230. This is the number of boosted trees trained.
- The random_state is set to 42. This can prevent too many combinations since there are 230 estimators. This will avoid taking too long of searching for the best tree.

For the other hyperparameters, they are set to the default value provided by XGBoost. Final setting:

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.05, max_delta_step=0, max_depth=5,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=230, n_jobs=0, num_parallel_tree=1, random_state=42,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

Figure 11. Setting of XGBoost

Figures below are the first tree and last tree generated.

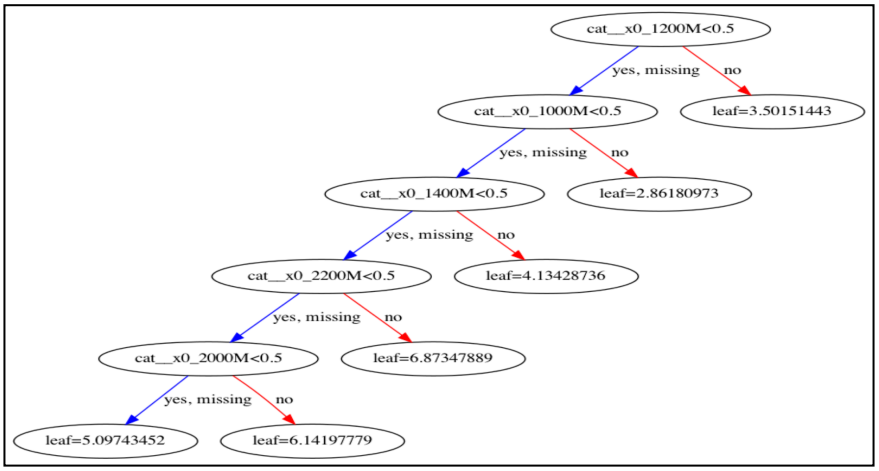

Figure 12. The first tree generated

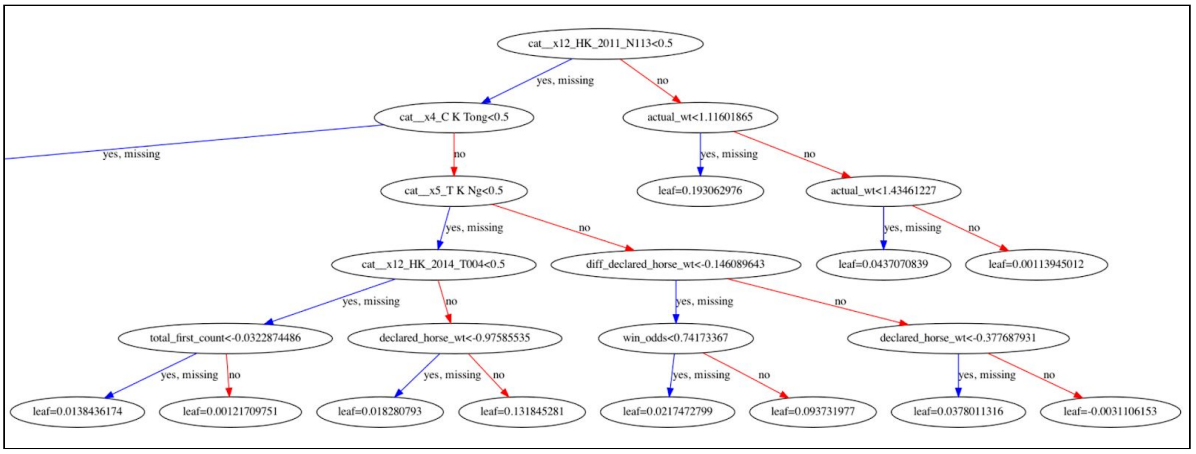

Figure 13. Part of the last tree generated

Then, using the trained model to predict the finishing time of the horses in 2019. Sort the finishing position in ascending order by the predicted finishing time and grouped by each race. Here is one of the predicted race:

	0	df_date	race_date	season	race_index	horse_id	race_no	class	win_odds_	total_race_count_	finish_time_sec	rating	plc
9856	99.330414	2019-12-29	2019/12/29	19/20	293	HK_2018_C197	10	Class 3	3.9	12	100.15	64.0	3
9849	99.410896	2019-12-29	2019/12/29	19/20	293	HK_2018_C443	10	Class 3	5.1	4	100.69	72.0	8
9851	99.426056	2019-12-29	2019/12/29	19/20	293	HK_2017_B023	10	Class 3	5.8	25	100.83	63.0	11
9845	99.607193	2019-12-29	2019/12/29	19/20	293	HK_2017_B189	10	Class 3	8.9	35	100.49	77.0	5
9858	99.632103	2019-12-29	2019/12/29	19/20	293	HK_2014_T098	10	Class 3	12.0	69	100.13	61.0	2
9854	99.680313	2019-12-29	2019/12/29	19/20	293	HK_2017_B161	10	Class 3	10.0	17	100.30	72.0	4
9855	99.829094	2019-12-29	2019/12/29	19/20	293	HK_2016_A193	10	Class 3	13.0	45	100.82	63.0	10
9852	99.874359	2019-12-29	2019/12/29	19/20	293	HK_2017_B317	10	Class 3	16.0	21	101.25	70.0	12
9847	99.891357	2019-12-29	2019/12/29	19/20	293	HK_2017_B353	10	Class 3	15.0	18	100.51	67.0	6
9857	100.045280	2019-12-29	2019/12/29	19/20	293	HK_2017_B203	10	Class 3	25.0	25	100.09	70.0	1
9848	100.065971	2019-12-29	2019/12/29	19/20	293	HK_2015_V338	10	Class 3	33.0	41	100.51	77.0	7
9850	100.530182	2019-12-29	2019/12/29	19/20	293	HK_2016_A127	10	Class 3	69.0	19	102.16	76.0	14
9853	100.536362	2019-12-29	2019/12/29	19/20	293	HK_2017_B330	10	Class 3	74.0	14	101.71	78.0	13
9846	100.768089	2019-12-29	2019/12/29	19/20	293	HK_2018_C489	10	Class 3	102.0	3	100.69	69.0	9

Figure 14. One of the predicted race[2]

[2] The column “0” is the predicted time.

3.3.2 Result

The result of XGBoost is satisfying. First, the first tree is meaningful:

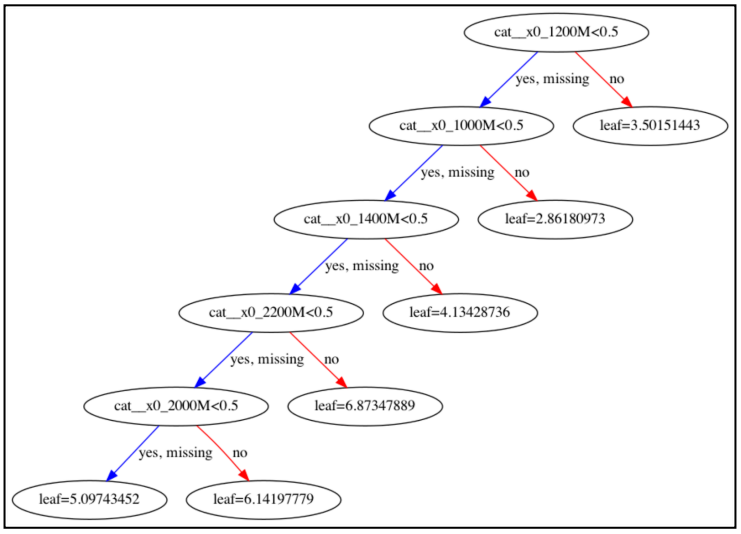

Figure 15. One of the trees generated

The model notices that the length of race determines the finishing time since 1000M's finishing time of a horse is obviously smaller than 1400M. The last tree is more complex than the first tree. That means the gradient boosting is working.

R2 score is 0.9974 This score means how the prediction data fits the actual data. 1 means the same as the actual data. 0 is completely different from the actual data. The R2 score is near to 1 so the prediction is working well.

The accuracy of predicting the first place is 30.3704% and the accuracy of predicting the first, second and third place is 7.1605%. The accuracy compared to the previous project is acceptable.

3.3.3 Simulation

For simplification, we choose only 1 betting type 'win', each time we bet 100 dollars to the first place (Win) according to the predicted data. Then find the result from the actual data. If we win, the cash balance will be added by 10* win_odds - 10. If we lose, the cash balance will be minus by 10.

All the graphs below are about the cash balance. Y-axis represents the cash balance. X-axis represents the race order by date.

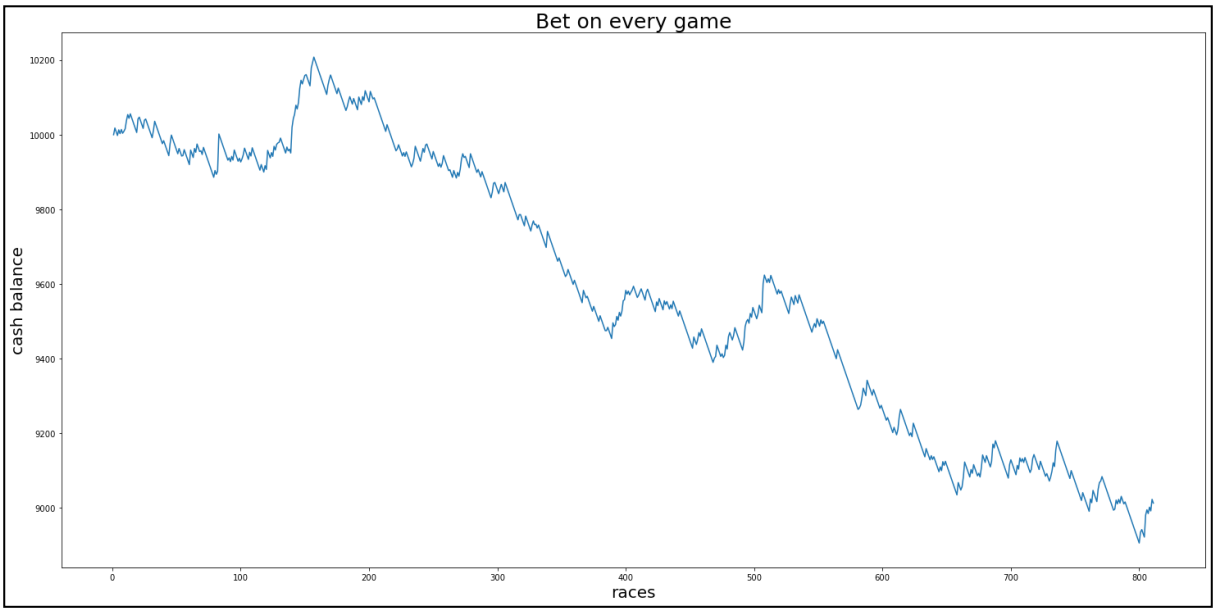

Figure 16. Cash balance when betting on every game

It is intuitive that if we just simply bet on every game, we will definitely have a negative gain. Where the last cash balance is 9013, we lose 1000 eventually.

So we try to find some criteria that we can obtain a positive gain which is ‘how many times the horse participated in horse racing before’

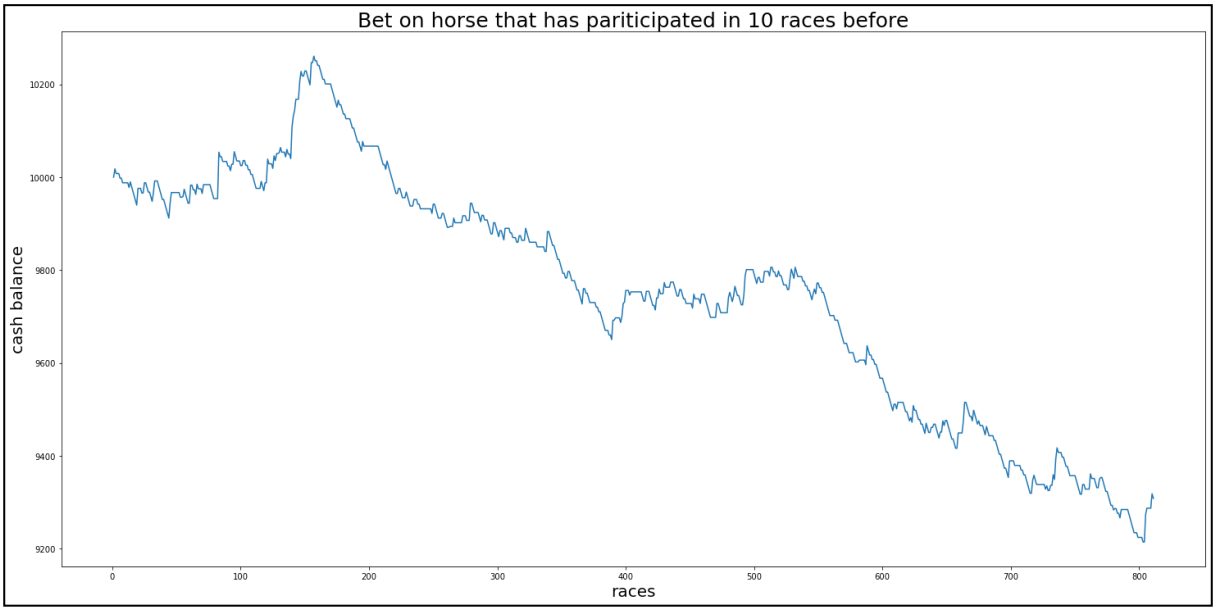

Figure 17. Cash balance when betting based on criteri: 10 races

If we increase the limitation, the cash balance is higher than before. The last cash balance is 9308.

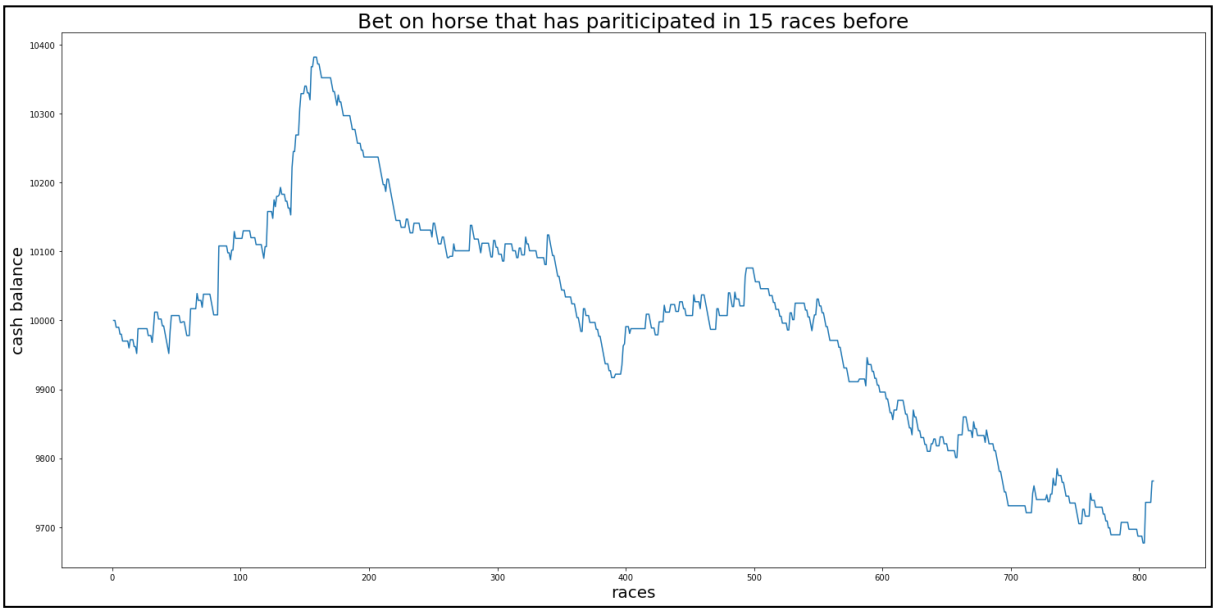

Figure 18. Cash balance when betting based on criteria: 15 races

Keep adding the limitation by 5. The last cash balance is now 9767.

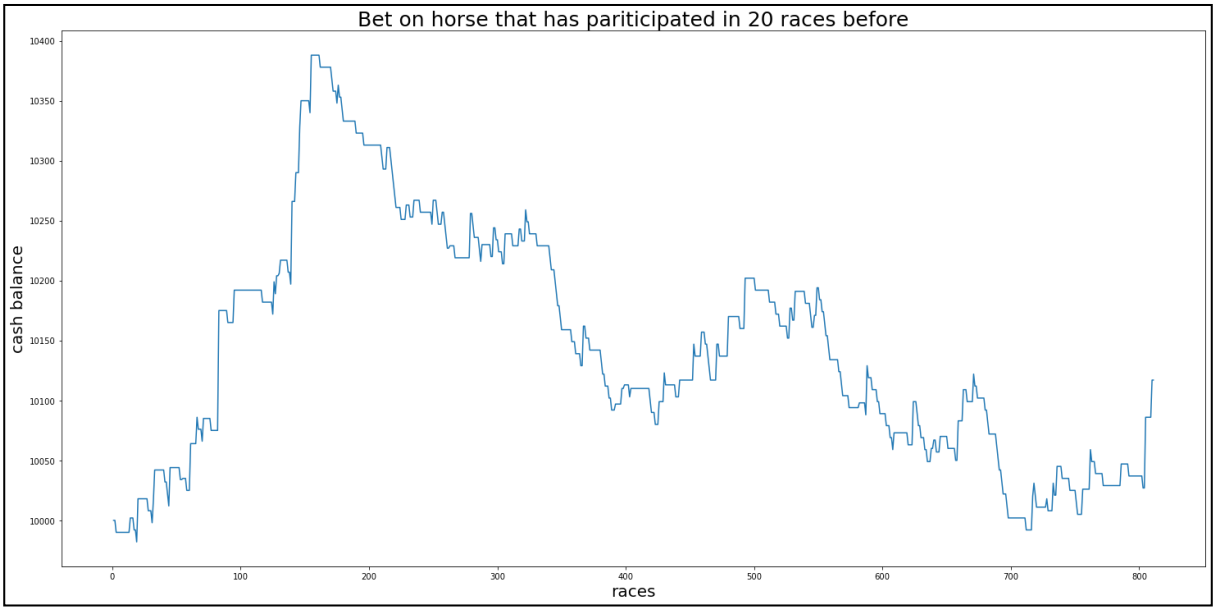

Figure 19. Cash balance when betting based on criteria: 20 races

Now the limitation is set to 20. The last cash balance is positive at 10117 and most of the time the cash balance is above 10000.

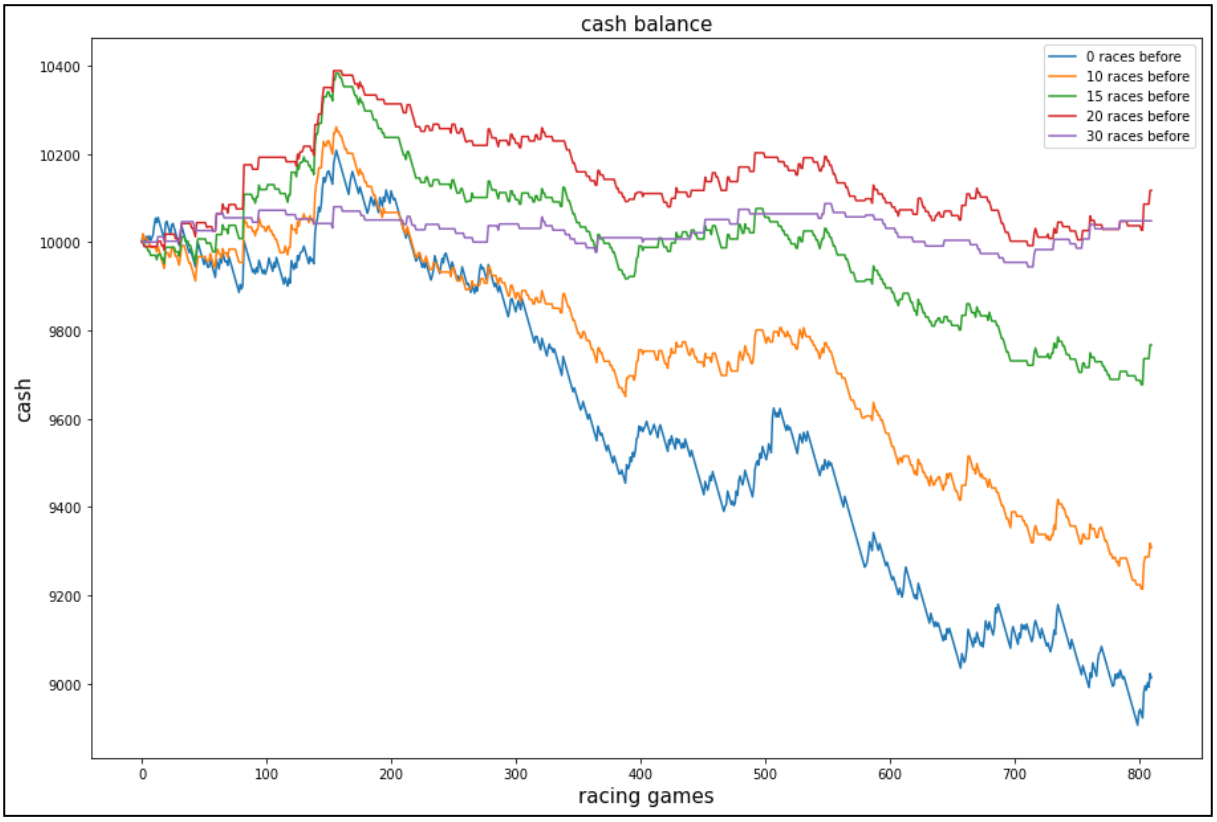

Figure 20. Cash balance when betting based on different criteria

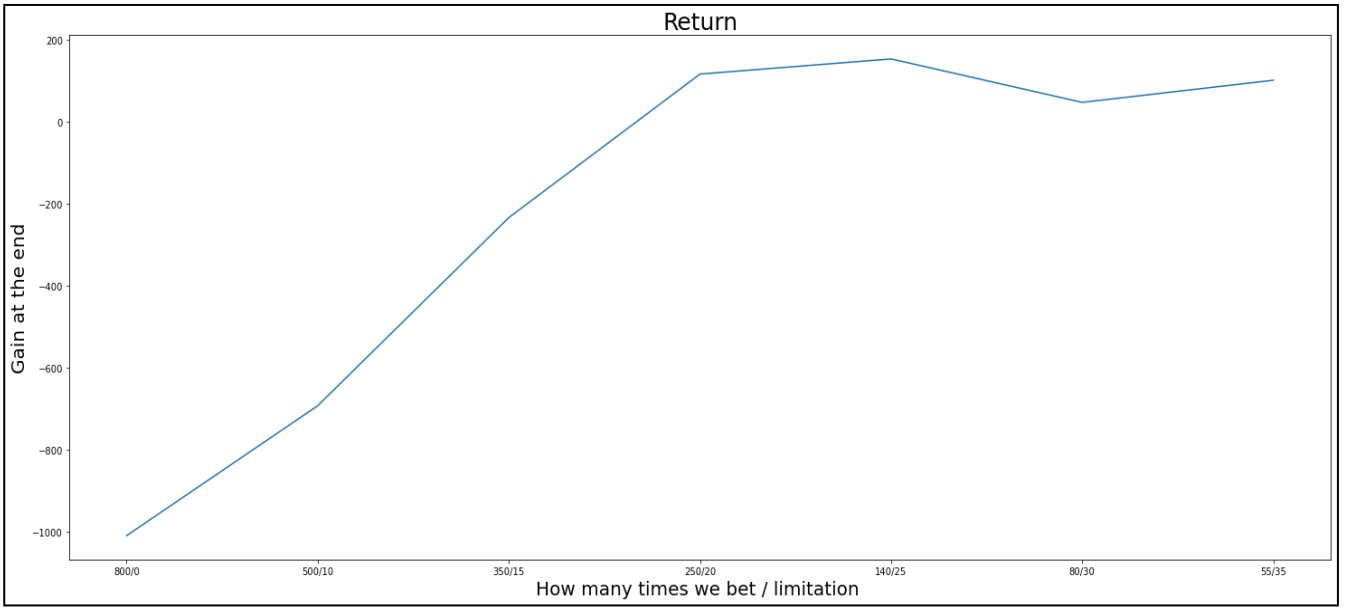

Figure 21. Return of betting based on different criteria

As shown above, the criteria of how many races the horse has participated is effective to decide whether to bet or not.

3.3.4 Conclusion and Analysis

3.3.4.1 Analysis of the results

- There are positive correlation between the ‘participation experience of horses’ to the ‘win rate’
    - The performance of the horses are more stable
    - These experienced horses are old and most of them belong to class 3,4,5.
        - The competition in these classes are not that keen

- Although we can use a criteria to decide whether to bet or not, but the return may not be stable and maximized, and the return is not high, which may not be suitable to use in real life, it can also be treated as a classification problem whether to bet this not or.

- If we just focus on the first place, it is nearly impossible for us to have a positive return, although we have a criteria to decide whether to bet or not.

- This prediction with 30% accuracy may help us in Reinforcement learning by ordering the input horses.
    - The place we predicted, will be used as the order of the horse we input to the reinforcement learning model.

- We learn how complicated horse racing betting is through Building this model.
    - Highly accurate prediction is not enough, we need a strategy to bet.

3.3.4.2 Conclusion

To conclude, The accuracy of the XGBoost is pretty good. However, it is not enough for us to just have a nice accuracy, it is more important that if we can find some betting strategies that help us to obtain positive gain. We will use the XGboost prediction to help construct the reinforcement learning. And find a better strategy to bet.

Chapter 4 Reinforcement Learning

4.1 Introduction

Reinforcement learning is a kind of machine learning that learns by interacting with the environment. The learner is not taught what actions to be taken but learns from the actions [9]. In our case, we do not want to teach the agent how to bet on horse racing. Instead, we want the agent to learn by himself and find the best way to bet.

4.1.1 Algorithm

The basic algorithm of reinforcement learning is shown in the figure below:

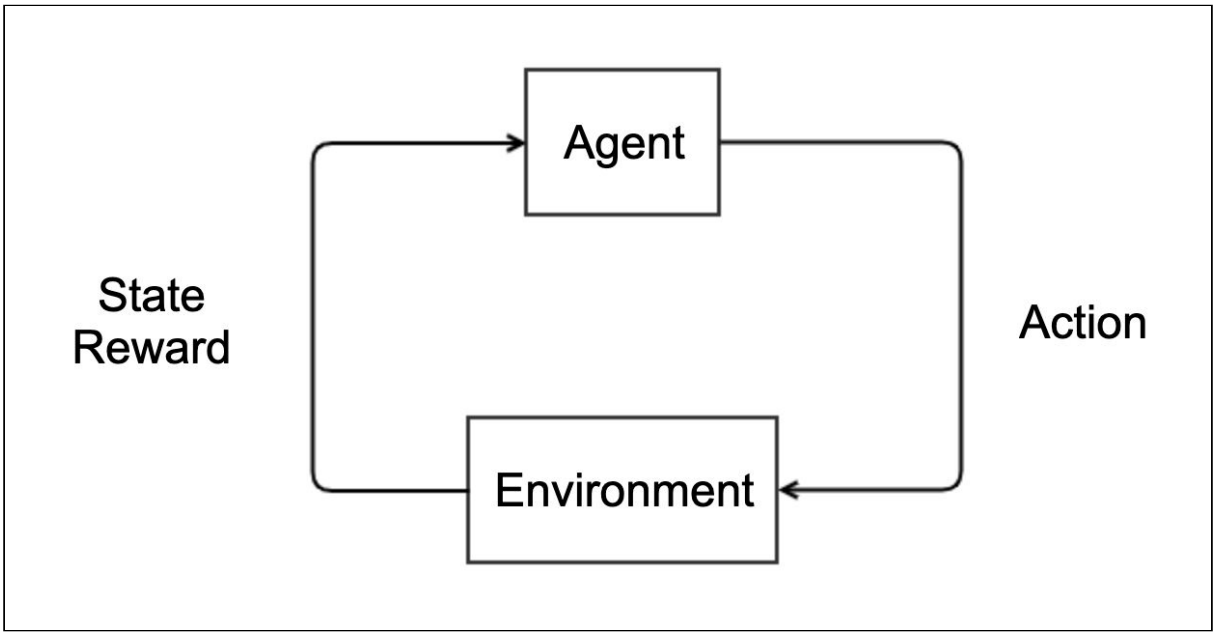

Figure 22. algorithm of RL

The agent which is trained interacts with the environment by action. The action will be based on different policies due to different RL algorithms. After the action, the environment will return a new state and reward to the agent. The agent will know its performance by the reward. Then the agent can update the policy of action by the reward.

4.1.1.1 Markov Decision Process

Formally, reinforcement learning problems can be modeled as Markov Decision Process:

- M = {S,A,O,T,Ɛ,r}
- S : State space containing a set of states (s) that the agent can be.
- A : Action space containing a set of actions (a) that the agent can make.
- O : Observation space containing observations (o) that the environment gives based on the agent's state
- T : Transition operator which is the probability of choosing the future state based on the current state ($p(S_{t+1}|S_t)$)
- Ɛ : Emission probability which is the probability of getting the observation based the state ($p(O_t|s_t)$)
- r : Reward function which is the reward from the action on the state. This can tell us which actions on which states are better ($r(s_t,a_t)$)

4.1.1.2 Goal

The goal of reinforcement learning can be written as:

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

Formula 8. Reinforcement learning

In words, the goal of reinforcement learning is to find the best policy $\theta^\star$ which can bring the best expected total reward:

$$E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

Formula 9. Best expected total reward

4.1.2 Environment

There are many kinds of environments based on the requirement. There are discrete environments and continuous environments which means finite states and infinite

states. More kinds of environments are episodic or non-episodic, single agent or multi agents.

In this project, there will be only one agent, finite states since we only train the agent to learn how to bet. For the environment, it will be an episodic environment which means the current action will not affect the next action. It is partially observable since the agent will not know which horse wins until it finishes its action.

4.1.3 Different Reinforcement Learning Algorithm

Due to different situations in reinforcement learning, Based on the basic algorithm, there are other types of reinforcement algorithms including policy gradients, value-based, actor-critic, model based.

- Policy gradient is a policy-based algorithm. The agent will take an action based on the policy. The policy is different actions with their probability.(Ex. Left 50%, Right 50%) The final reward of each episode will improve the policy.

- Value based means the agent will take an action based on the reward after an action. Most likely, the reward will be given immediately after the action is taken.

- Actor-critic combines policy gradient and value-based. Critic is a value based network while actor is the policy gradient.

- Model based requires a base policy (model). The agent will act based on a built model and improve the model based on the reward.

4.1.4 Objective

In this project, we are using a value-based algorithm called deep Q-learning. The reason for us to choose value based is value based algorithm is more similar to the situation of gambling. We will place a bet on a horse because we believe that that horse will give us reward (money). Same reason, the agent will place a bet on a horse because the horse will give the agent reward. We hope to find out the best policy to get the best profit from horse racing.

4.2 Q-learning

Q learning is a value based method which learns from the value. Although we are not using Q learning in horse racing, it is the basis of deep Q-learning. The reason for not using Q learning will be mentioned at the end of this part.

4.2.1 Q function in a table

Definition of Q function: Q(s,a) For s = state, a = action, The value of Q(s,a) will be updated after each step of the episode.

If we group all the Q functions together, it can be represented as a Q table:

State \ Action	a1	a2
s1	value1	value2
s2	value3	value4

4.2.2 Algorithm

The algorithm of Q learning is to update the Q table and based on the Q table to choose the best action [17].

1. Initialize Q function Q(s,a) to some random values
2. Take an action from a state using epsilon-greedy policy from Q function
3. Observe the reward and the new state

4. Update the Q table by :

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma maxQ(s'a') - Q(s,a))$$

Formula 10. Q-learning

5. Repeat step 2 to step 4 until terminal state

Epsilon greedy policy means that the action will be picked up by the best value. There is a parameter called Epsilon which decides the probability of choosing. Using the table above as the example, let epsilon be 0.9 and value1 > value2. If the agent is in the state s1, there will be 90% of choosing action a1 and 10% of choosing a random action.

For the equation:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma maxQ(s'a') - Q(s,a))$$

Formula 10. Q-learning

- α is the learning rate to decide how many differences need to be learned. This is useful for convergence.

- r is the reward obtained from the action.

- γ is the discount of the future reward. The discount is to decrease the effect of the future reward to the current reward so the agent can learn to take the other action not just because of the future reward. We want a balance between the future reward and the current reward.

MaxQ(s',a') means the next state's best value of the Q function. This is the future reward.

The whole equation is for updating the Q table by calculating the difference between the actual value and the estimated value. The difference will be minimized to build the best Q table to find the best result for best efficiency.

4.2.3 Example

We chose a simple game “Cartpole” from OpenAI gym. The goal of the game is to prevent the pole falling over the cart.

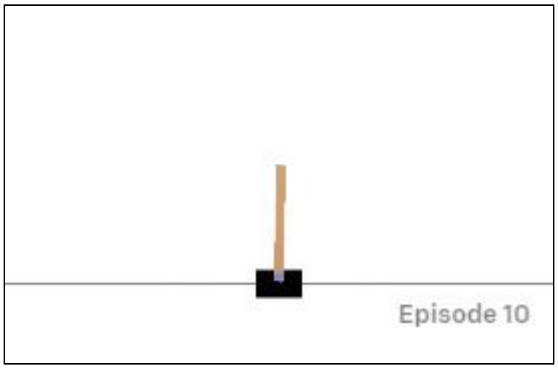

Figure 23. Capture of ‘Cartpole’

This game is just a little experiment of Q learning and we want to know how many episodes are needed to finish its learning. This value will be a reference to our project to apply reinforcement learning to horse racing.

In this little experiment, the reward will become larger as longer the pole staying on the cart.

4.2.3.1 Convergency

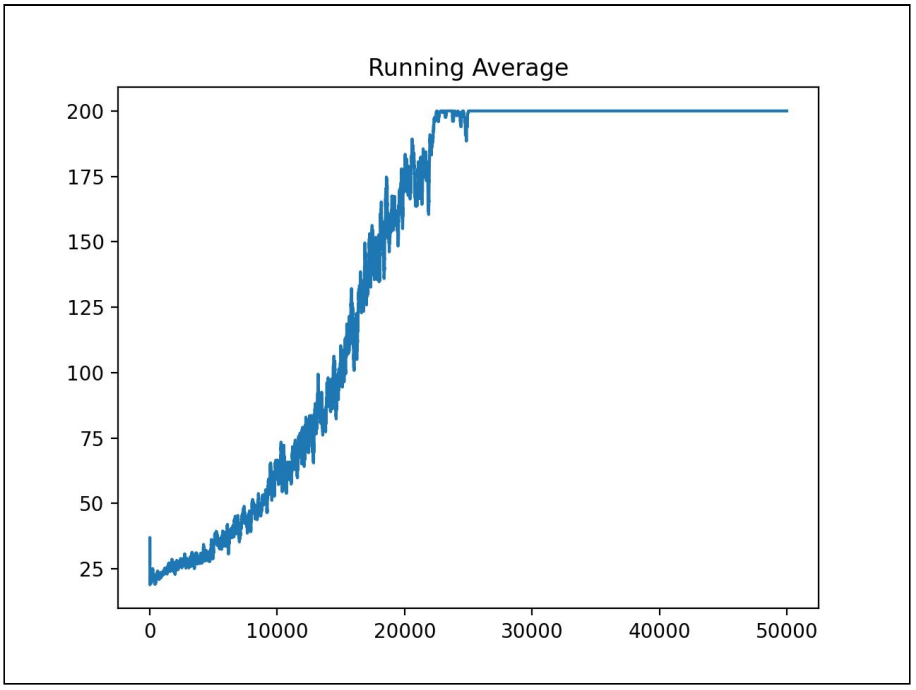

Figure 24. Reward average

Here, the x-axis is the episodes and the y-axis is the reward. We can see that to gain the maximum reward, the agent needs 25000 episodes to finish the training to get a stable reward.

If we apply this result to horse racing, comparing the complexity of horse racing to this game, it will need much more episodes and time to finish its training to make the result stable.

4.2.4 Reason not using Q-learning

Q-learning usually is used in some simple problems or games. Using it in horse racing will cause some troubles.

First, a lot of horses' data will cause a huge Q table. There will be a lot of states and for each state, there are 15 actions (Buy horse a, b, c, … ,or skip). The efficiency will be very low. It is too difficult to complete the Q table.

Second, although we go through all states and build a Q table to store all states and actions, that only means that the agent remembers all races' results and bet based on that result. When the agent meets a new race with new horses, it will take random action only.

Therefore, we will use the other value based method, Deep Q learning, which will be mentioned in the next part.

4.3 Deep Q-Learning with MLP policy

In this section, we apply reinforcement learning in horse racing. In order to update the $Q(s, a)$ function, we need to iterate through all the state. However, in real life, the state space is too complex for the agent to go through all of the states. So, we need an approximate q-value for those similar states. In deep Q-Learning, we use a neural network to approximate the $Q(s, a)$ function, by combining the Q-learning and deep neural network.

4.3.1 Deep Q-Network

Deep Q-network Architecture:

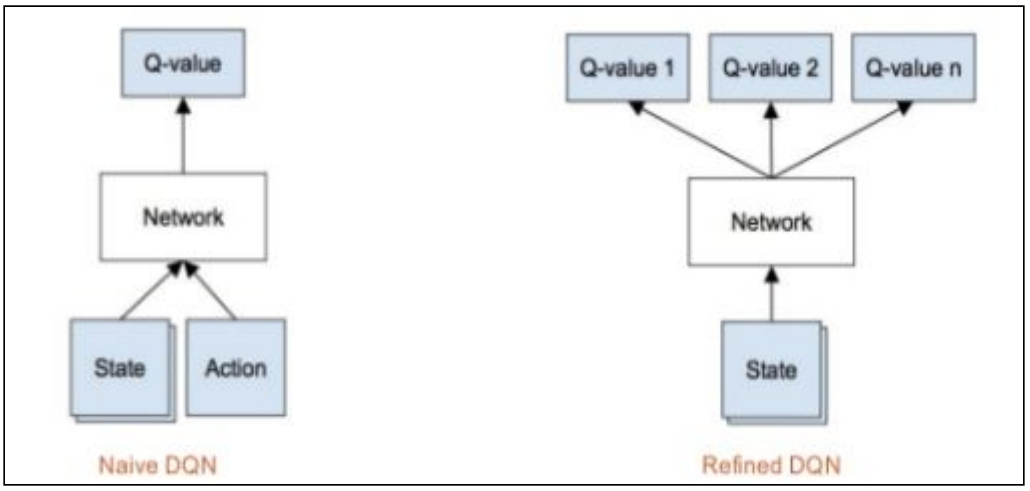

Figure 25. Deep Q-network Architecture

In this project we will choose the Refined DQN.

4.3.1.1 DQN Loss function

$$E_{s,a,r,s'}[((r+\gamma \max_{a'} Q(s',a';\theta')) - Q(s,a;\theta))^2]$$

Formula 11. DQN Loss function

The above loss function is used to find the approximate q-value in the network[19], If the θ' is a constant, we can use gradient descent on the loss function respect to θ.

4.3.1.2 Moving target problem

$(r+\gamma \max_{a'} Q(s',a';\theta'))$ is the target network, and $Q(s,a;\theta)$ is the prediction. However, these two terms are correlated, so if we change θ frequently, the θ' will change frequently too, our network is training for a moving target which will lead to diverge.

So one of the solution from DeepMind[2] states that θ' is updated with a long enough interval so that we can avoid it from diverge.

4.3.1.3 Experience Replay

Q-learning always uses the reward of the last state to train the value network, but the optimization techniques required that the data is independent and identically

distributed sampling, so we store an amount of the latest step into the replay buffer, it performs the gradient descent on a sample minibatch. So the correlation between the data can be canceled out [2].

Chapter 5 Applying DQN in horse racing

In this semester, we try a simple approach, directly giving all the feature values of all the candidates in a single game, to see if the agent can bet on a ‘winning’ horse. Several betting types of horse racing are mentioned in the first chapter, However, due to simplicity, we concern only one betting type ‘A bet to win’ (if the horse wins the first place, you win) in this semester.

5.1 Construction

We use the Stable Baselines3 and OpenAI Gym[9] to construct our DQN reinforcement learning.

5.1.1 Environment

We have 10000 cash balance, and if the agent wins, we give him 10 * win odds cash. If he loses, we take 10 cash from him. That means the agent can only bet 10 dollars.

5.1.2 Observation Space

As mentioned before, the feature values of all the horses and the cash balance are the observation space to the agent. Also the win odds of the horses are given. Since there are horse racing games which less than 14 horses participate in, we would set all the feature values of those invalid horses to be -99.

5.1.3 Action Space

We simply want the agent to choose whether to bet and which to bet. So there will be a total 15 actions {‘1’, ‘2’, … ‘14’, ‘15’} that an agent can choose, where action {‘1’, ‘2’, … ‘14’} represent to bet on the number 1, 2, … ,14 horse respectively. And action {‘15’} represents not to bet.

5.1.4 Step

After the agent goes through one games, The agent will receive first two major informations

1. The new game state : feature values of all 14 horses of the next game and the cash balance after the last game.
2. The reward obtained based on the last action.
3. Game over or not.
    a. Once the agent loses more than 1000 dollars where the cash balance < 9000, then it is game over.
    b. Once it goes through all the racing games, it is game over too.

5.1.5 Reset

In order to train the agent to learn which horse is nice to bet, other than recognizing the winning horses in each state, the state jumps to random racing games everytime we reset, and the cumulative reward will be reset.

5.1.6 Termination state

In horse racing games, we said it is game over when the agent loses more than 1000 dollars or when it goes through all the games.

5.2 Reward Function and discount factor

Reward for the action guides the agent to learn. It is the only feedback that the agent will receive, It is extremely important for the reinforcement learning algorithm to perform well. A badly constructed reward function will lead to an unwanted behaviour of the agent. Also a good reward function can let the agent learn faster.

In a horse racing game, the basic idea for the reward function is simple, when it wins, it receives a large reward. otherwise, it receives a penalty. Intuitively, the agent will not bet on any racing game if we only give reward or penalty on these two

situations, as it will not receive any penalty. So we will give a penalty which is less than the 'bet but lose' situation.

5.2.1 Idea of Reward Function

Just simply give penalty or reward based on the change of cash balance:

- $R(bet\ and\ win) = C_1 * \Delta Cash\ Balance\ ,\ where\ C_1 > 0$
- $R(bet\ but\ lose) = C_2 * \Delta Cash\ Balance\ ,\ where\ C_2 < 0$
- $R(\ do\ not\ bet\ ) = C_3 * win\ odds\ of\ the\ true\ first\ place,\ where\ C_2 < C_3 < 0$

Recall that there are invalid horses in the input, so we have to deal with the case when the agent chooses to bet on a invalid horse. There are few approaches [20]:

1. Ignore it
    a. $R(bet\ on\ invalid\ horse) = 0$
2. Same penalty as 'do no bet'
    a. $R(bet\ on\ invalid\ horse) = R(do\ not\ bet)$
3. Large penalty
    a. $R(bet\ on\ invalid\ horse) = C_4 < C_3$

A paper [20] claims that giving a penalty to the agent is one of the common ways, and masking is one of the methods to do so. However, We will not mask 'invalid horse' to 'valid horse' so we directly give it a reward which same as 'do not bet'

We believe that if we choose the first option, it is not reasonable as it becomes 'do not bet' without any penalty. Therefore, the agent may always choose this action as there is no penalty.

For the third option, the penalty of 'bet on a invalid horse' is large, the agent will tend to bet on those small number horses to avoid getting a large penalty, it probably will just choose the first five actions: 'bet on horse 1', 'bet on horse 2' … 'bet on horse 5' . Since there are at least 5 horses in every race, these actions will never be invalid action.

5.2.2 Discount factor

The formula below shows the total future reward G on the current step t is based on all the rewards multiply γ in all the future steps:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Formula 12. Total future reward with discount factor

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T,$$

Formula 13. Total future reward without discount factor

Where the γ is the discount factor, the agent will 'treat' the future reward more important if γ is closer to 1 [24].

Learning how to bet on horse racing games one by one is an episodic environment, each separated horse racing game will end in the same way. Rewards are given according to the result, which are 'bet and win', 'bet but lose' and 'dont bet'.

In a racing game, the agent is supposed to learn how to win all the bets, the future bet is as important as the current bet. Also, T is a finite number as in our environment, so $\gamma = 1$ which is the second way, the agent should treat every game at the same level.

5.3 DQN with MLP Policy

5.3.1 Network Structure

There are two hidden layers with 32 neurons each and using the tanh activation function, 15 neurons at the output layer.

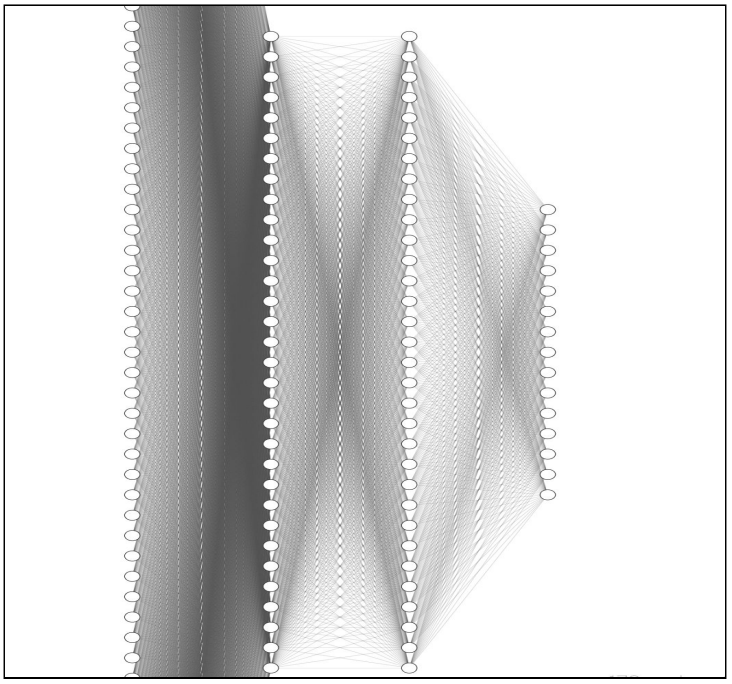

Figure 26. Network structure

5.3.2 Input data structure

There are 14 horses in a race, each horse contains 3963 features which all those features are the same as we used in XGboost, so there are (14 x 3963) inputs in each state.

Intuitively, the order of the horses will affect how the agent acts.We choose to use the prediction of XGBoost to order the input, from slowest to the fastest. All the invalid horses will be ordered as the fastest horse. The reason is that we want to help the agent to identify which horses are worth to bet.

5.4 Training and Testing

5.4.1 Number of candidates

There are 3080 races in our training set.

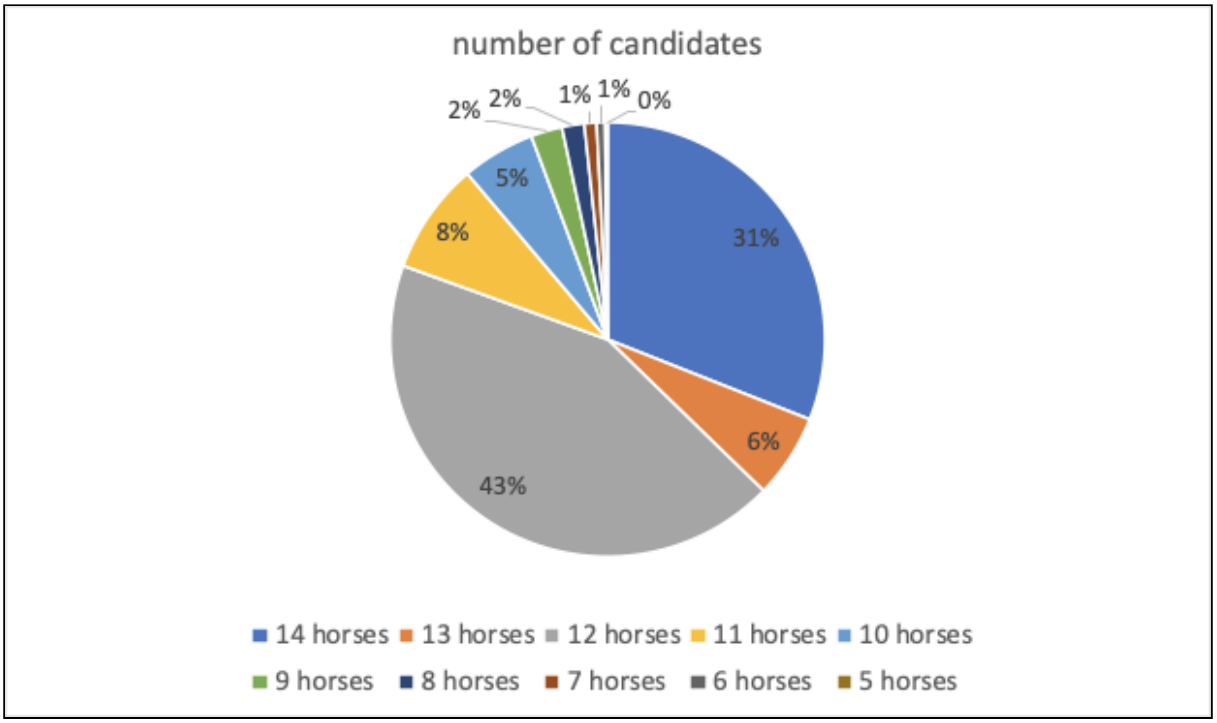

Figure 27. number of candidates [3]

There are 12 or 14 candidates in 74% of all the races. Which is the major of the horse racing games.

5.4.2 Results of reward function

Recall the reward function: $R(bet\ on\ invalid\ horse) = R(do\ not\ bet)$

As the reward of 'bet on invalid horse' and 'do not bet' is the same, so the agent will treat both are the same. Assume there are 14 valid horses, if the agent does not want to bet, it should choose 'do not bet'.

Since reinforcement learning is learning how to play in a specific env, we will also provide the learning result in a training phrase, and we let the agent to bet on the testing set after training.

[3] The exact number is shown in Appendix

5.4.2.1 Convergence of reward

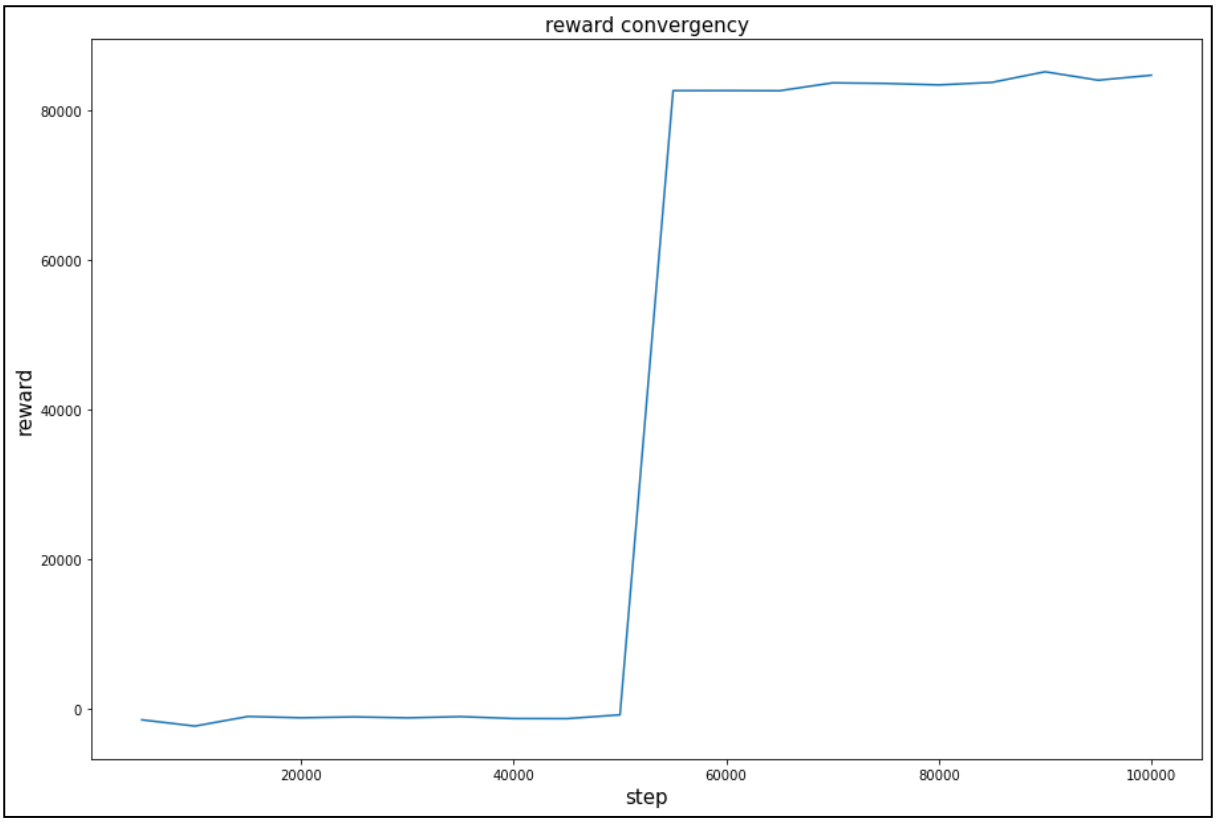

Figure 28. Convergence of DQN

The figure shows that after around 58000 steps which is around 75 episodes in training, the reward converges to 80000 reward.

5.4.2.2 How the agent bet in training set

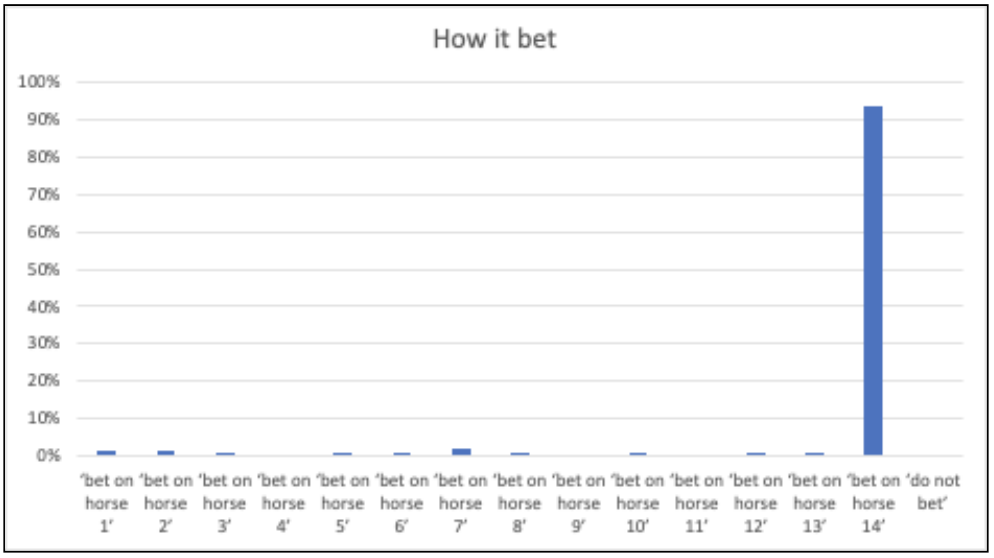

Figure 29. How the agent bet in training set[4]

[4] Recall that horse n is referred to the input order.

The agent chose to bet on horse 14 most of the time in the 3080 races. And there is a large difference, where 94% of action is ‘bet on horse 14’ and 0% in ‘do not bet’, ‘bet on horse 8’, ‘bet on horse 9’, ‘bet on horse 10’ and ‘bet on horse 12’. The remaining action occupies around 4%.

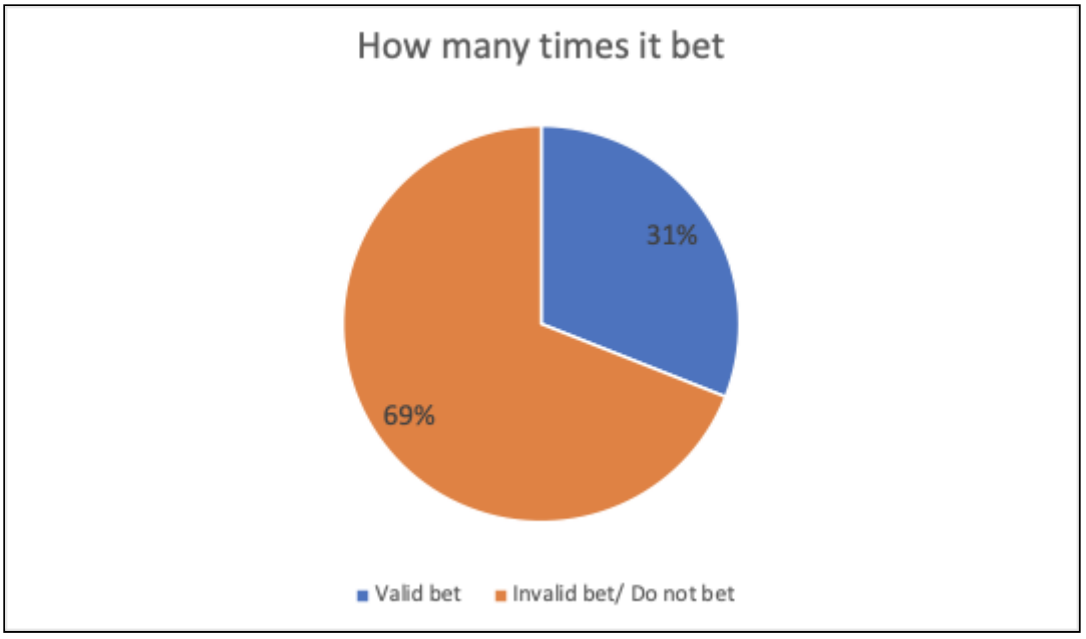

Figure 30. How many times the agent bet in training set

Since there are invalid horses, ‘bet on horse’ does not indicate how many times it is bet. It bet only 31% of 3080 races. And there are 70% invalid bets/’Do not bet’. And the ‘bet on horse’

5.4.2.3 How the agent bet in testing set

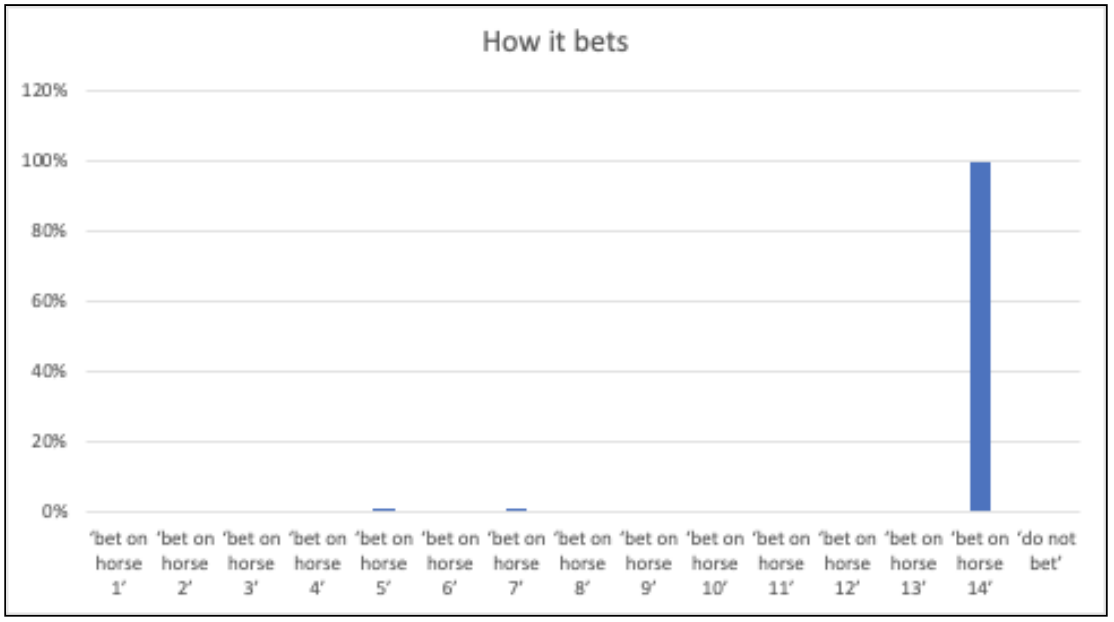

Figure 31. How the agent bet in testing set

The agent chooses to bet on the horse 14 most of the time in 810 races, where 99% of action is ‘bet on horse 14’.

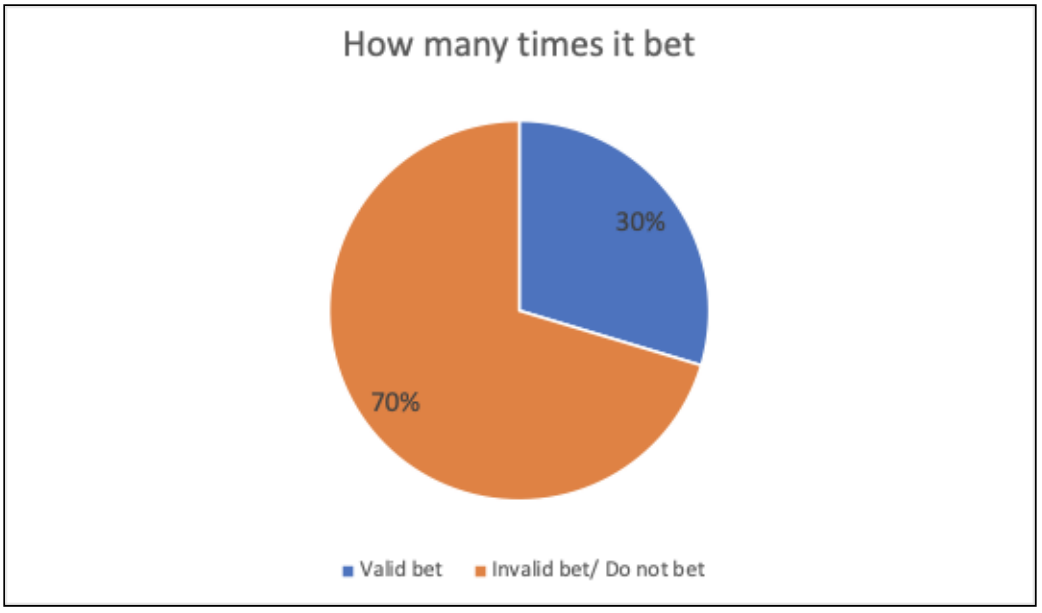

Figure 32. How many times the agent bet in testing set

Since there are invalid horses, ‘bet on horse’ does not indicate how many times it is bet. The agent bet on 239 games in 810 games. Which is nearly 30%, the ratio is close to the ratio of training sets.

5.4.2.4 Win or lose in training set

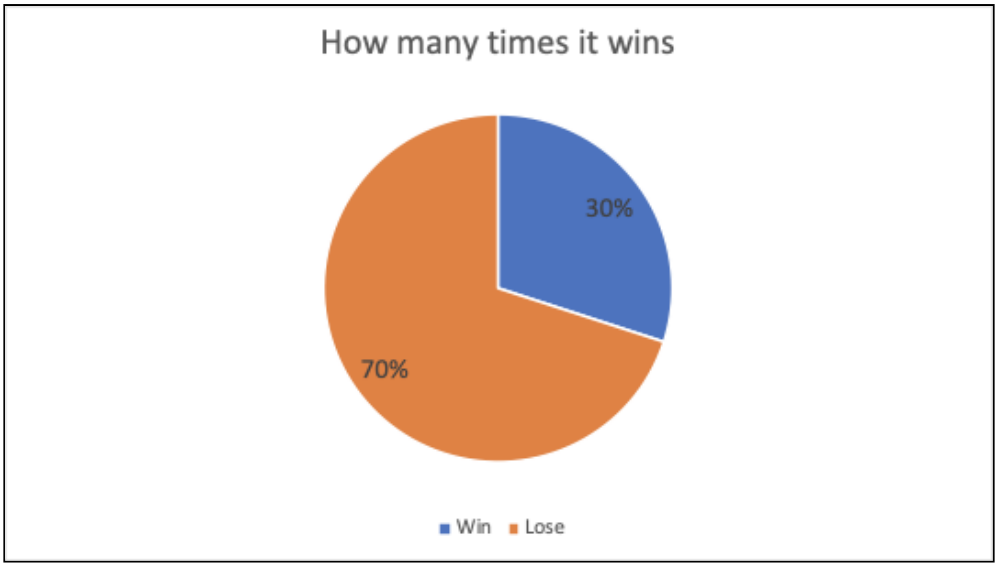

Figure 33. Win rate of the agent in training set

In 3080 racing games, the agent bet on 31% of games with 30% wins and 70% losses.

5.4.2.5 Win or lose in testing set

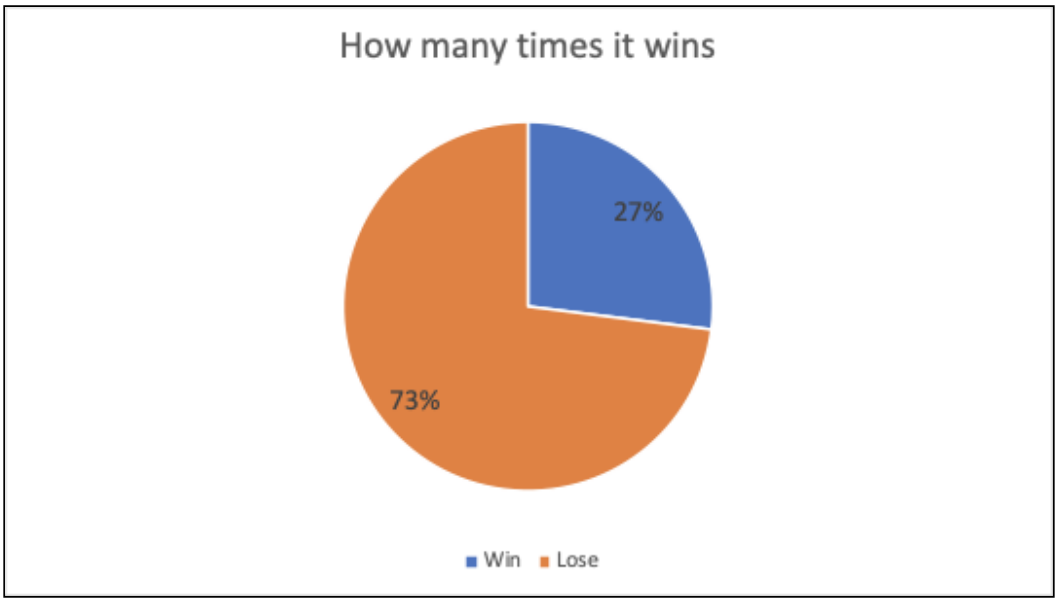

Figure 34. Win rate of the agent in testing set

In 810 racing games, the agent bet on 70% of games with 27% wins and 73% losses.

5.4.2.6 Cash balance in Training Set

Recall that the agent can only bet with 10 dollars.

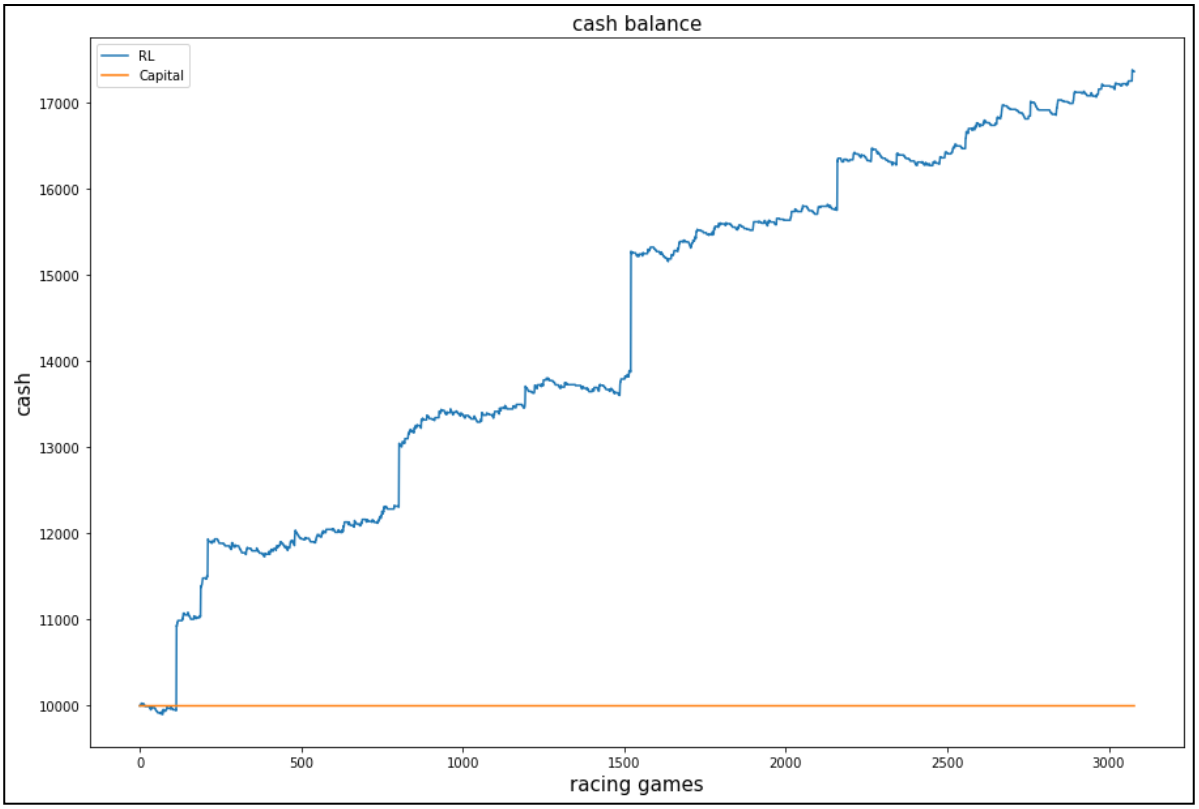

Figure 35. Cash balance in training set

Although we can only win 30% of the games, But the agent will sometimes bet on horse with higher win odds, there is some vertical lines in the graph, the cash balance is increased rapidly, the largest gain is 10*140 (betting amount* win odds) - 10(betting amount). The cash balance is increasing and it reaches around 17000 at the end.

5.4.2.7 Cash balance in testing Set

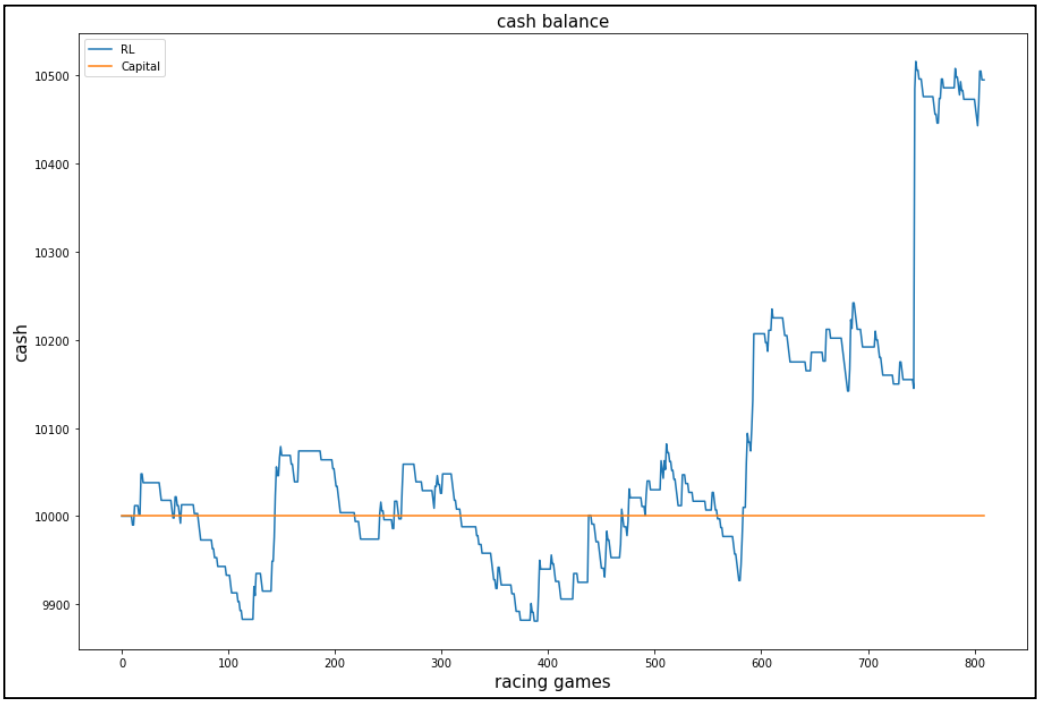

Figure 36. cash balance in testing set

The cash balance is around 10000 before the first 600 games, and after the agent wins some games with higher win odds, it rapidly increases near the 600th race and 750 race. The largest gain is 10*34(betting amount* win odds) - 10(betting amount).

5.5 Result Analysis

The agent's behaviour is developed from our rewarded function. There are some interesting behaviours that we observed.

5.5.1 Analysis on the agent behaviours

- As we can see that most of the time the agent bet on horse 14
    - Since horse 14 is most likely the horse to win where the accuracy is 30% from XGBoost.
    - It is the most likely path that the agent can receive the largest reward.
    - If there are some horses with large win odds, it tries to bet on those horses to gain a larger reward.

- 0 ‘do not bet’ action
    - The reward from the ‘do not bet’ and ‘invalid horses’ is the same, the agent will treat it the same action.
    - The agent bet on the fastest horse predicted in all the games with 14 horses, which is reasonable as that horse is likely to win with 30%
    - It can avoid the ‘do not bet’ penalty by betting all the horse 14.

- When the horse with high win odds, the agent will try to bet on those horses as it will return a greater reward to the agent.
    - Since the reward is based on how much it wins, when there are horses with high win odds, the reward of betting on those horses will be higher than just betting on the most safe action ‘bet on horse 14’.

- Bet on the Horse racing game with 14 horses only
    - As the penalty of selecting ‘invalid horse’ is less than ‘losing’, the agent may choose invalid horse other than betting on ‘invalid horse’ in most of the cases.
    - In order to deal with this problem, we think there are multiple approaches
        - Improve the reward function
        - deal with the invalid horse input format
    - It will be much easier to improve the reward function. So we try this approach.

Although the cash balance is increasing so far, it is increasing because of large jumps. In a very long run, we can’t ensure such a large jump and it may go down eventually.

5.6 Improvement of reward function

Although there are positive gains from the previous setting, the agent only bet on the games with 14 horses. The reason is that the reward of trying to bet on races with different numbers of horses is likely to be less, as there are 90% of races with 1 invalid horse. To increase the reward we can:

1. Increase the reward of ‘winning’
2. Decrease the penalty of ‘losing’
3. Decrease the penalty of ‘invalid betting’

The third option is not suitable as we don 't want it to be a cheaper version of 'do not bet’.

Recall that

- $R(bet\ and\ win) = C_1 * \Delta Cash\ Balance\ ,\ where\ C_1 > 0$
- $R(bet\ but\ lose) = C_2 * \Delta Cash\ Balance\ ,\ where\ C_2 < 0$
- $R(\ do\ not\ bet\ ) = C_3 * win\ odds\ of\ the\ first\ place,\ where\ C_2 < C_3 < 0$

The parameter we need to tune is C1 and C2.

5.6.1 Tuning the parameter

After our testing, we find a set of parameters that let the agent not only bet on races with 14 horses. However, the candidate number of races is limited to 13 and 14. The agent bet on 193 races with 13 horses races, 955 races with 14 horses races. The agent bet on all the games with 13 and 14 horses, there is no filtering.

5.6.2 Comparison between reward functions

We train the agent based on the new reward function and compare it with the agent with the old reward function and see if the agent behaved differently.

5.6.2.1 Convergence on reward

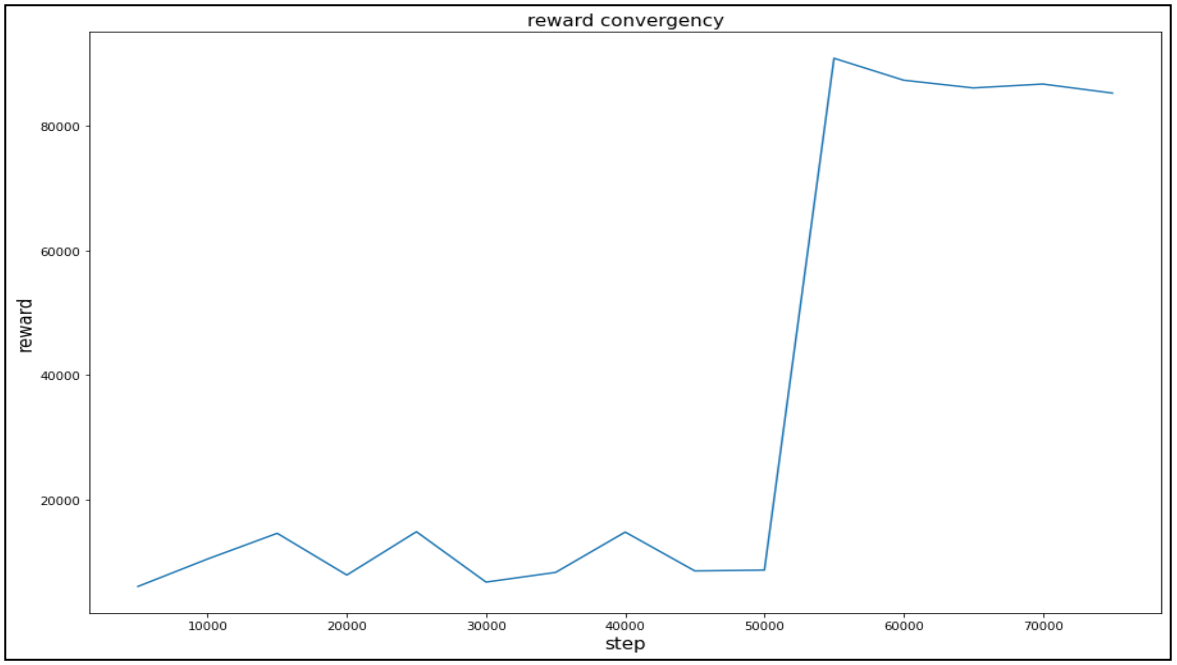

Figure 37. Convergence of DQN with new reward function

The agent with the newly constructed reward function converges after around 60000 steps which is 70 episodes.

5.6.2.2 How the agent bets in training set

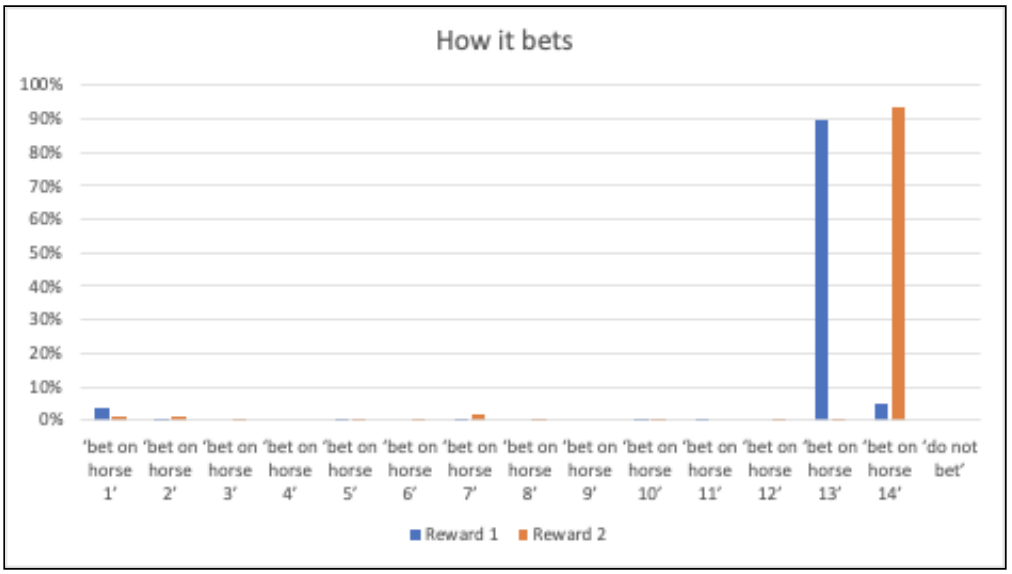

Figure 38. Comparison of 'how agent bet' between new and old reward function in training set

The agent bet most of the time on horse 13 after the improvement, there are 90% or 2769 times of action is 'bet on horse 13', and the time of 'bet on horse 14' action is decreased by 85%. The behaviour of the agent changed.

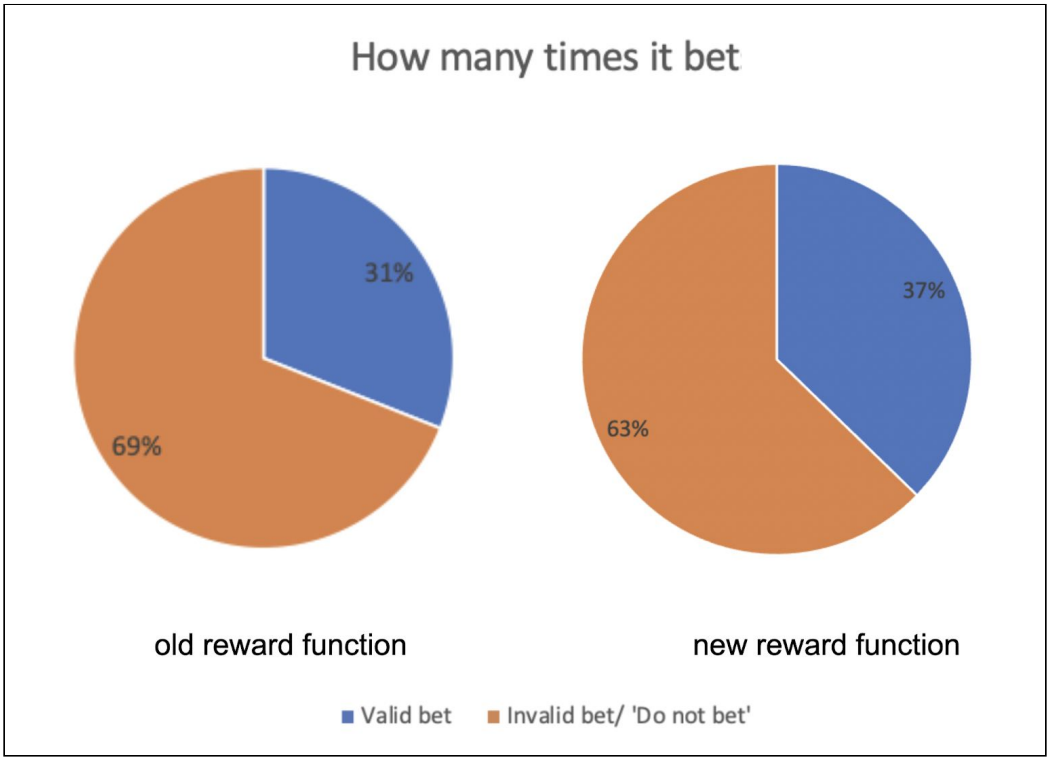

Figure 39. Comparison of 'how many times it bet' between new and old reward function in training set

The new agent bet more than before, The betting amount increased from 31% to 37%.

5.6.2.3 How the agent bets in testing set

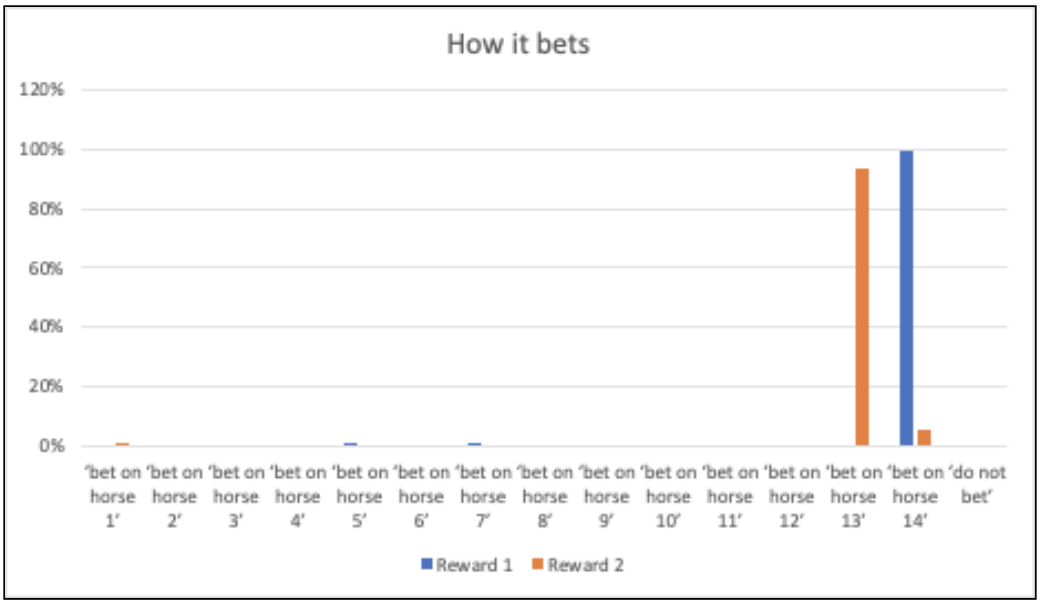

Figure 40. Comparison of 'how agent bet' between new and old reward function in testing set

In total 810 races, the agent chose 'bet on horse 13' 94% of times,while 'bet on horse 13' 5%, and 1% with those remaining action. And the old agent chose 'horse 14' most of the time. The behaviour of the agent changed.

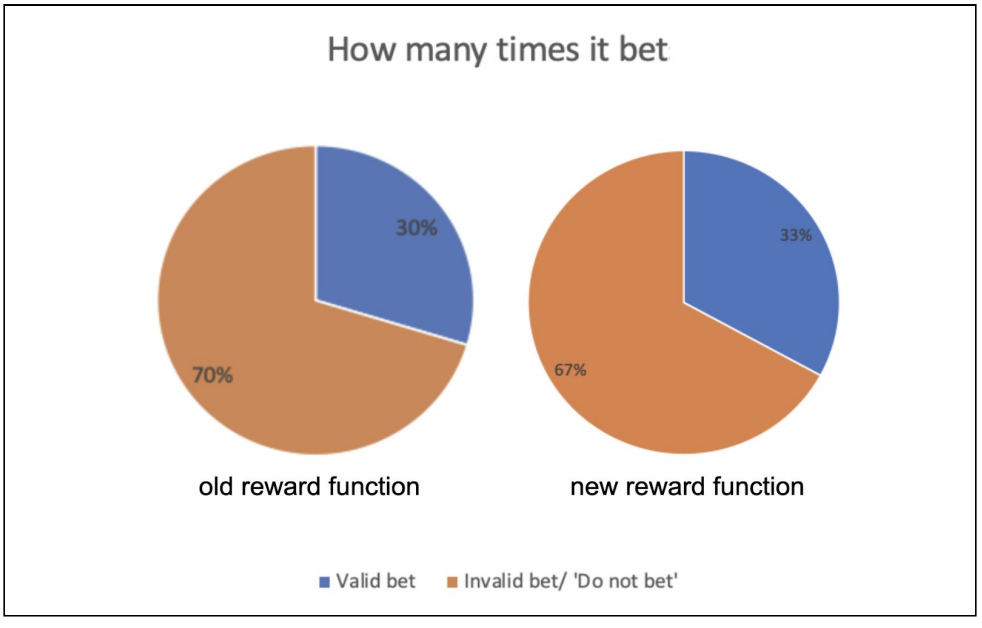

Figure 41. Comparison of 'how many times it bet' between new and old reward function in testing set

The new agent bet 3 more percent on the testing set than the old agent.

5.6.2.4 Win or lose in training set

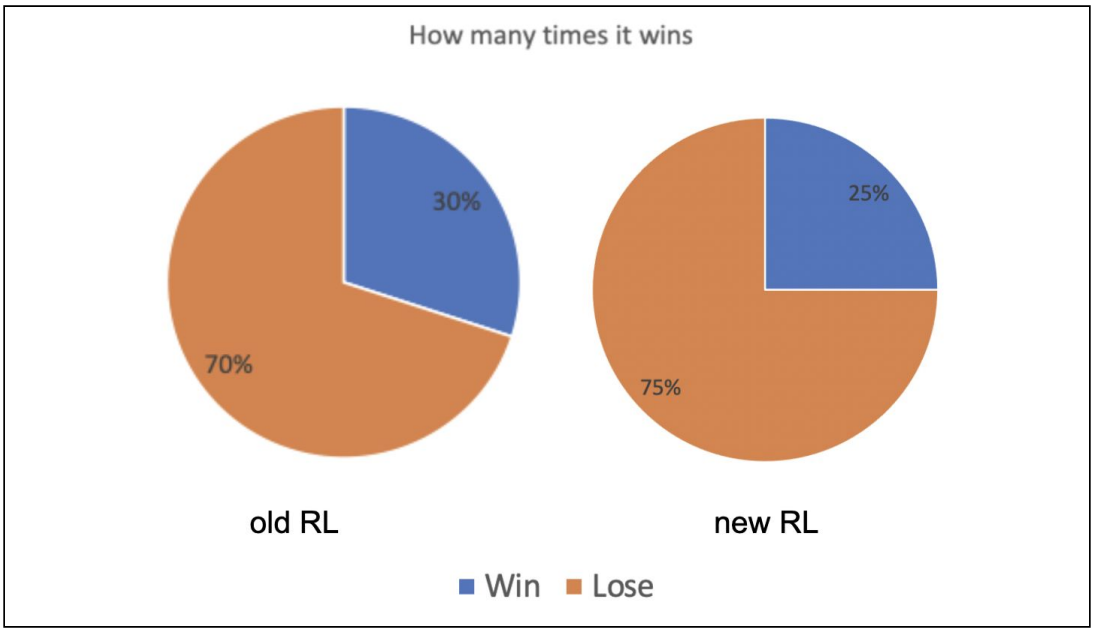

Figure 42. Comparison of win rate between the agent with new and old reward function in training set

The win rate has decreased from 30% to 25%.

5.6.2.5 Win or lose in testing set

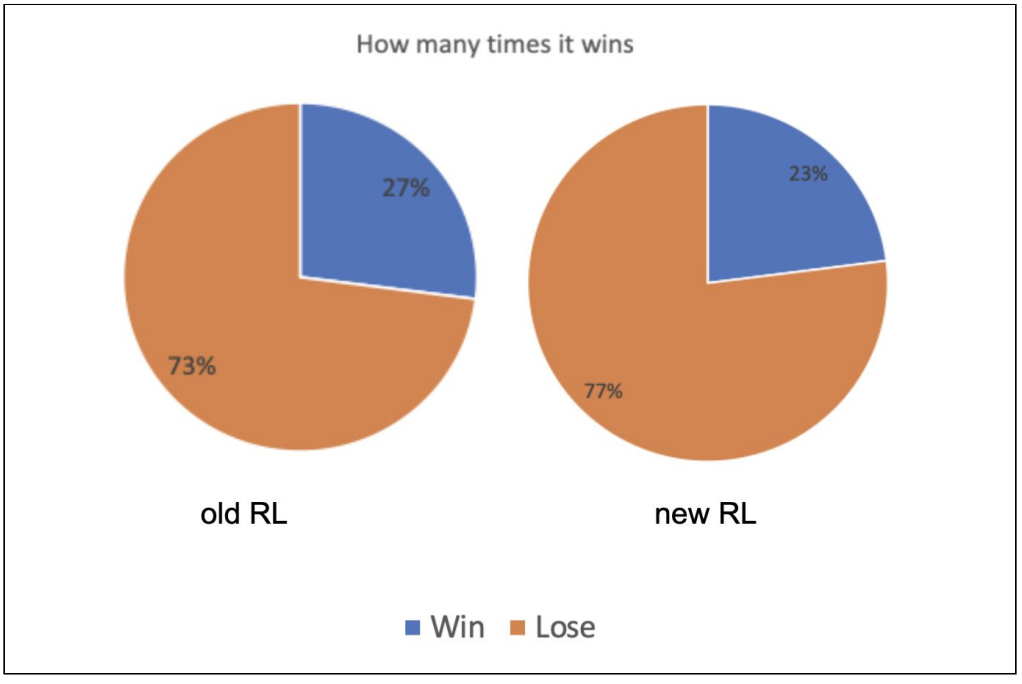

Figure 43. Comparison of win rate between the agent with new and old reward function in testing set

The win rate has decreased from 27% to 23%.

5.6.2.6 Cash balance in training set

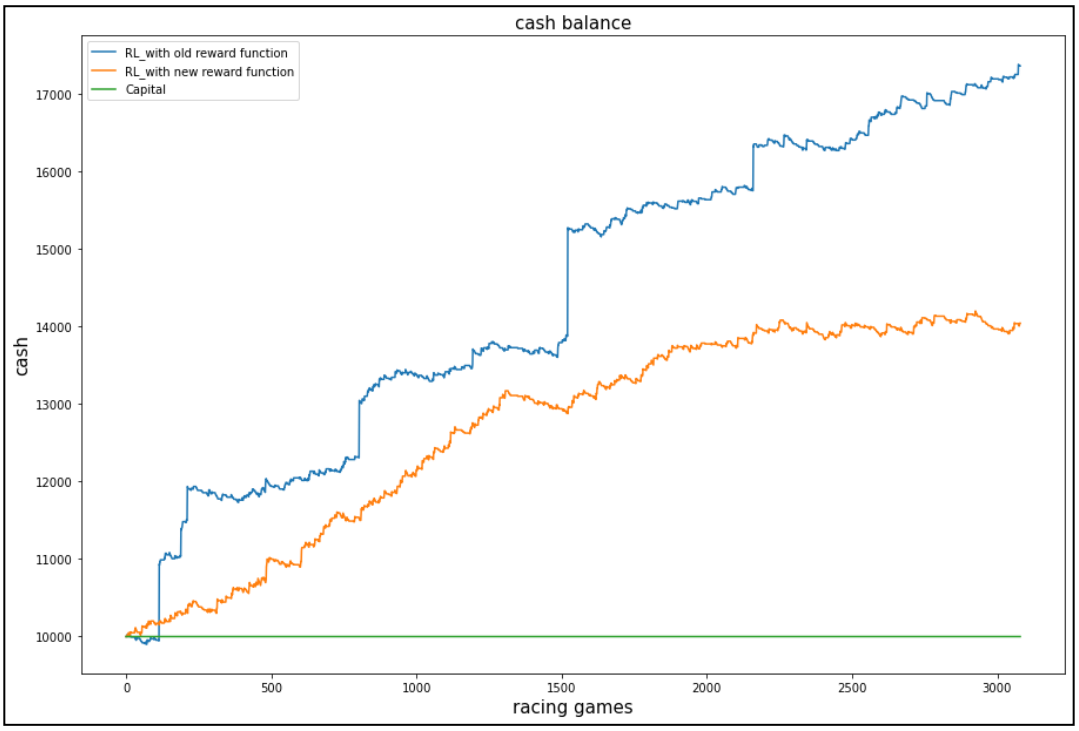

Figure 44. Cash balance between the agent with new and old reward function in training set

It is clear that the cash balance curve of the agent with new reward functions does not jump largely in the graph, that means the new agent doesn't bet on those horses with large win odds or as it may choose to bet on more races other than betting on the horse with large win odds. There are around 3500$ differences at the end of the games.

5.6.2.7 Cash balance in Testing Set

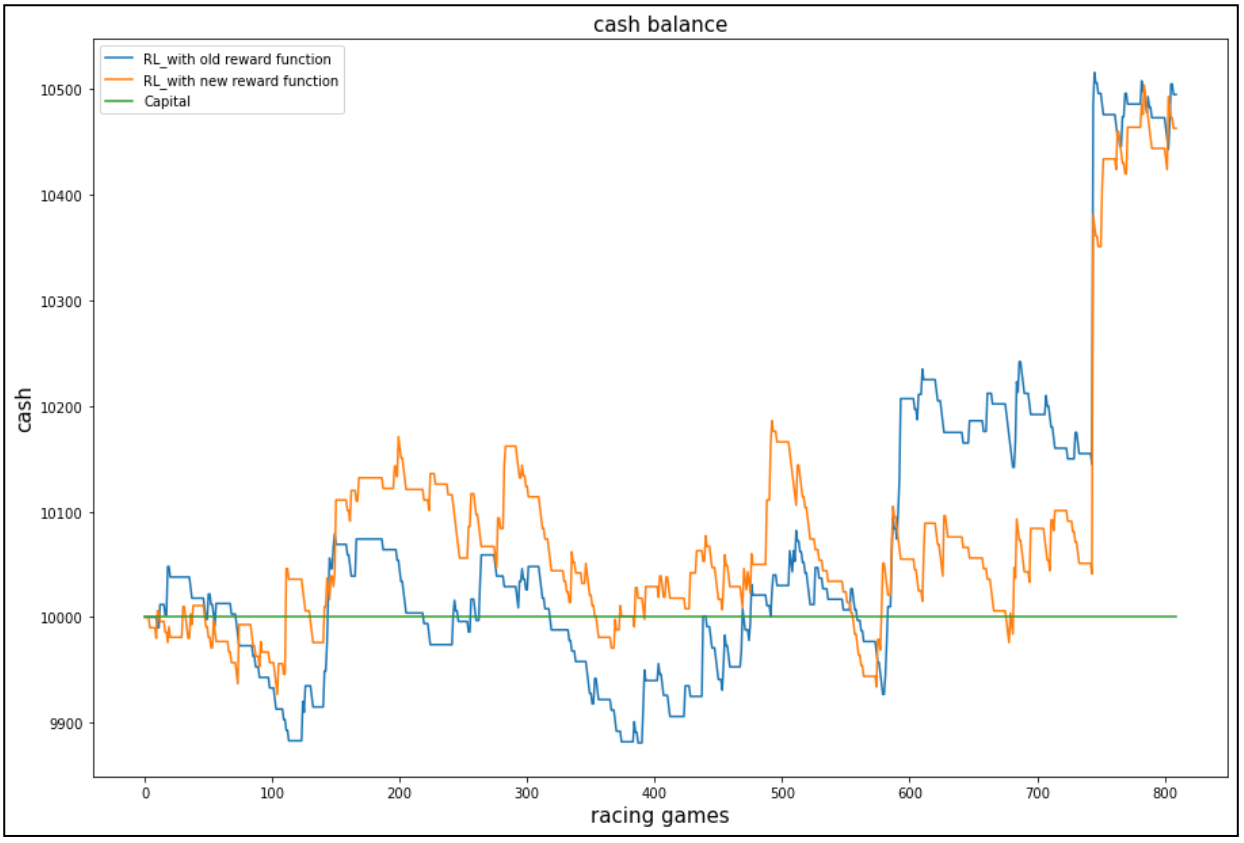

Figure 45. Cash balance between the agent with new and old reward function in testing set

The old agent performs worse than the new agent in the first 600 games, the new agent can remain a positive return longer, as the old agent tends to bet on a horse with large win odds and the new agent. And after some betting on higher win odds, the returns of the old agent is higher than the new agent.

5.6.3 Analysis and conclusion

Since we want the agent not just betting on races with 14 horses, we tune the parameter in the reward function. We increase the reward of ‘winning’ and decrease the penalty of ‘losing’ .

However, the reward function is still not perfect enough to let the agent to bet on all the races. The new agent just bet on races with 13 and 14 horses.

5.6.3.1 Result analysis

- The new agent chose the action ‘bet on horse 13’ most of the time instead of ‘bet on horse 14’
    - The ‘valid bet’ action is increased.
    - As the reward of ‘bet on invalid horse’ is less than before.
    - We need to modify the penalty of losing and the reward of winning again till perfect.
        - It is time-consuming and inefficient, so we may need to construct a new RL model or new input format of invalid horses.

- The new agent is more aggressive than the old agent
    - As we increase the reward of ‘winning’ and decrease the penalty of ‘losing’, it tries to bet more. So the new betting ratio is increased by 3% from the old one.

- The cash balance of the new agent at the end is less than the old one.
    - We believe that since the old agent only bet on races with 14 horses, it focused on learning how to win most of the races only on races with 14 horses, so it performed much better than the old agent.

- Win rate has decreased.
    - It is expected as we encourage the horse to bet on more games, as the reward of losing is decreased.
    - However, the result is also near what we expected as it is around 30%.

- The cash balance is not larger than before no matter in training and testing set
    - Although the agent with the new reward function tends to be more aggressive on betting more games, the agent is not aggressive on betting the higher win odds horses. As we can see in the cash balance graph, the cash balance never jumps rapid.

From these observations, We may want to set up a reinforcement learning model specifically, such as only races with 12 horses, other than putting all of the races in one single reinforcement learning model. In order to win more, we should also consider not just betting on more races, but also the win odds.

5.6.3.2 Conclusion

There are both positive cash balances till the end of the races in the training set and testing set, It is a bit out of our expectation that how the agents bet. But there are many imperfect ways such that the agent just bet on races with 14 and 13 horses, as the reward function is not constructed well, although the agent can bet well in these races. As we fixed the amount that the agent can bet, we can’t maximize the gain.

Chapter 6 Conclusion and Improvement

6.1 Betting strategy

The cash balances based on different betting methods are shown in the following graph.

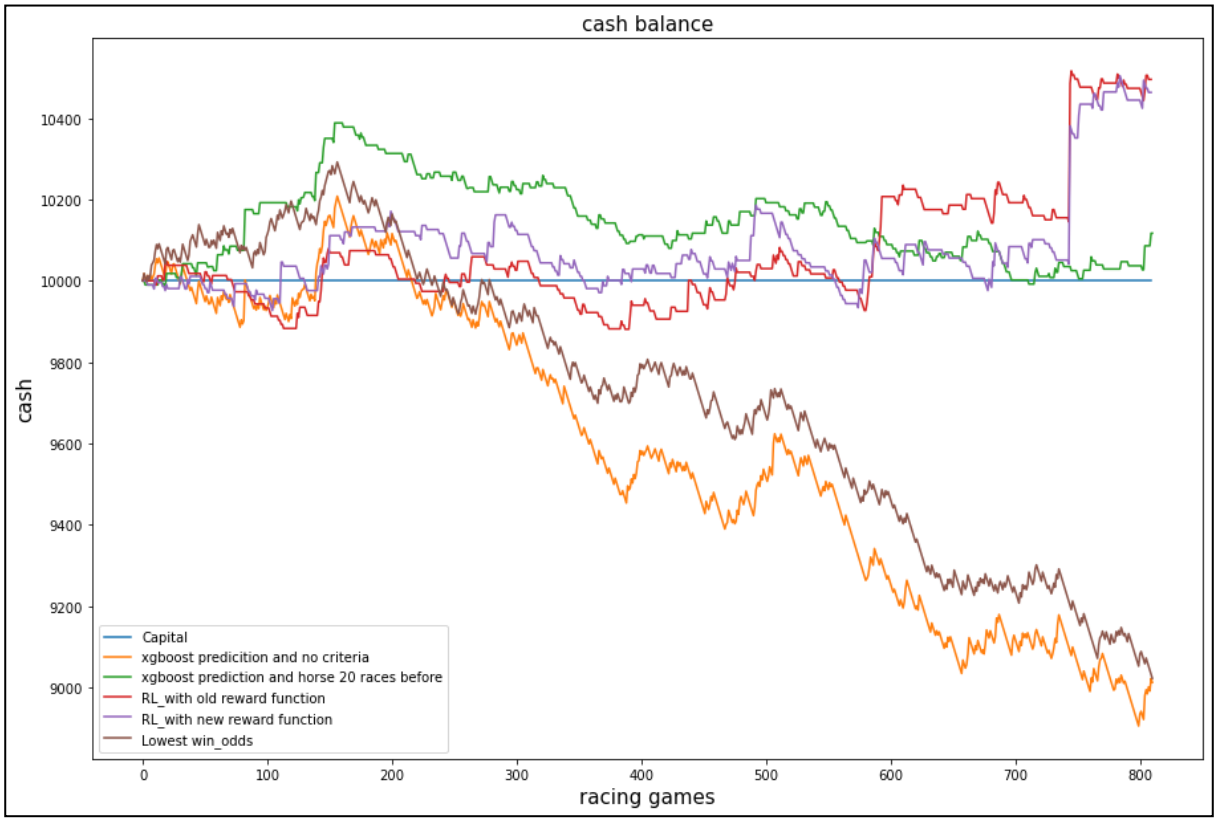

Figure 46. Cash balance of different betting strategy

We can see that only two betting methods return positive gain > 10000.

1. XGboost prediction with at least 20 times the horse has been participated before'
2. Use Reinforcement learning

In real life, it is extremely hard for us to find a criteria to bet or not, and most of the time, we are concerned about the horses in the first place we predicted. We seldom consider to bet on the horses that the model predicts is in the last place.

In RL, the agent will sometimes bet on the horses with lower ranking predicted with a large odds ratio. It may be possible to get a better return if we combine these models.

In this semester, we just use a simple betting method, 1 type of betting and fixed amount of betting. However, It will be much harder if we consider the betting types and how much to bet. These questions are suitable for reinforcement learning to deal with.

6.2 Problem encountered

There are several problems that we need to deal with and since this kind of application in reinforcement learning is not well developed.

6.2.1 Invalid horse input in Reinforcement Learning

Since the input order of the horses affect how the agent acts. The agent seldom bet on the races with less than 14 horses, as it wants to avoid the ‘invalid bet’. So we need to put more effort on how to deal with the invalid horse problem, maybe from the input format, and focus on a fixed amount of candidate races.

We have tried to tune the reward function, however it is time-consuming and not easy to achieve, so we may need to put effort in the other way.

6.2.2 Construction of reward function

Since the agent is sensitive to the reward function, it is hard for us to train it again and again as the training is time-consuming, we have to construct the reward function in a better way other than just using the trial and error.

We have to consider more situations, not just the three we considered in this semester, such as when the cash balance is high enough, then the penalty of losing should be less than the cash balance at a low level.

6.3 Future

Since we only use XGBoost and Deep Q Network, we still have not developed other models of reinforcement learning. There may be some models providing better results. As mentioned before, in this semester we use a simple betting method, 1 type of betting (Win) and fixed amount of betting. These will be considered in the next semester.

6.3.1 More betting types

In the introduction, there are multiple betting types like “Place” (Pick any one of the first three horses). We will train the agent to learn the other type of betting. We expect the accuracy of choosing “Place” should be higher since there are three horses who will give the reward to the agent.

The other types like “Quinella” which requires picking multiple horses will be applied to the model. These betting types will be more difficult to train and the accuracy is supposed to be lower but the reward will be much larger compared to “Win” or “Place”. The agent needs to act multiple times to place a bet. The odds’ calculation will also be different.

Also, multiple games betting (Pick multiple horses from multiple races) will be applied in reinforcement learning. For example, the agent will choose 1 horse from the first race, then choose 1 horse from the second race. The agent needs to act multiple times to place a bet. The odds’ calculation will also be different.

6.3.2 Different betting amount

In the first semester, we only train the agent to bet 10 dollars each. In reality, the gambler can choose the amount of betting by himself. This means there will be infinity of choice. However, we cannot put infinite actions into the action table since this is impossible. Therefore, we will classify the betting amount to a small amount (Ex. $10) and a large amount (Ex. $1000). We will see how the agent chooses from

these two choices. The last choice we want to see is “All in” which means to bet all the money it gets. We shall see the result in the next semester.

6.3.3 More models

In the first semester, we only apply Deep Q Network to horse racing. Models like policy gradient or actor-critic have not been used. Policy gradient which is a policy based reinforcement learning. We will train the probability of choosing a horse based on the reward gained. Then combining policy gradient and DQN, the actor-critic will be used. These two models will be used in the future and we will compare the result to the present result.

Also, the DQN can deal with the discrete output which is not suitable in our cases, so we have to use another algorithm such as DDPG.

Reference

[1] Autopilot AI. (n.d.). Retrieved July 22, 2020, from https://www.tesla.com/autopilotAI

[2] AlphaGo: The story so far. (n.d.). Retrieved August 2, 2020, from https://deepmind.com/research/case-studies/alphago-the-story-so-far

[3] Cheng, T., & Lay, M. (2017). *Predicting Horse Racing Result Using TensorFlow* (Rep.).

[4] Yide, L. (2018). *Predicting Horse Racing Result with Machine Learning* (Rep.).

[5] Wong, Y. (2019). *Horse Racing Prediction using Deep Probabilistic Programming with Python and PyTorch* (Rep.).

[6] Wong, K. (2020, November 24). A Brief History of Horse Racing in Hong Kong. Retrieved November 29, 2020, from https://discovery.cathaypacific.com/brief-history-hong-kong-horse-racing/

[7] HKJC. (2020). Performance Highlight 19-20. Retrieved 17 July, 2020, from https://corporate.hkjc.com/corporate/common/chinese/pdf/report-2019-20/HKJC_AR20_Book_A_Highlights.pdf

[8] HKJC (n.d.). Racing Academy. Retrieved July 29, 2020, from https://entertainment.hkjc.com/entertainment/english/learn-racing/racing-academy.aspx

[9] Ravichandiran, S. (2018). *Hands-on reinforcement learning with Python: Master reinforcement and deep reinforcement learning using OpenAI Gym and TensorFlow*. Birmingham, UK: Packt Publishing.

[10] The Hong Kong Jockey Club. (n.d.). Retrieved July 29, 2020, from https://www.hkjc.com/home/english/index.aspx

[11] Hong Kong Observatory. (n.d.). Retrieved July 29, 2020, from https://www.hko.gov.hk/en/index.html

[12] Zychlinski, S. (2018). [Web log post]. Retrieved from https://towardsdatascience.com/the-search-for-categorical-correlation-a1cf7f1888c9

[13] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

[14] Kearns, M. (1988). Thoughts on hypothesis boosting. *Unpublished manuscript*, *45*, 105.

[15] Wade, C. (2020). *Hands-on gradient boosting with XGBoost and scikit-learn: Perform accessible machine learning and extreme gradient boosting with python*. Birmingham: Packt Publishing.

[16] Friedman, J. H., Hastie, T., & Tibshirani, R. J. (1998). *Additive logistic regression: A statistical view of boosting*. Stanford (CA): Stanford University. Division of Biostatistics.

[17] Winder, P. W. P. (2020). *Reinforcement Learning*. Van Duuren Media.

[18] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping”, in ICML, vol. 99, 1999, pp. 278–287.

[19] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529-533.

[20] Huang, S., & Ontañón, S. (2020). A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*.

[21] DeepSenseAI (n.d.). Retrieved September 29, 2020, from https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

[22] Sewak, Mohit (2019) "*Deep Reinforcement Learning: Frontiers of Artificial Intelligence"* Springer Singapore Pte. Limited

[23] Graesser, Laura ; Keng, Wah (2019) "*Foundations of Deep Reinforcement Learning: Theory and Practice in Python"* Addison-Wesley Professional 2019

[24] Dayan, Peter ; Balleine, Bernard W "*Reward, Motivation, and Reinforcement Learning*" Neuron (Cambridge, Mass.), 2002-10, Vol.36 (2), p.285-298

Appendix

There are 3010 races in our testing set:

number of candidates	number of races
14	955
13	193
12	1332
11	256
10	168
9	72
8	51
7	28
6	19
5	6

There are 810 races in our testing set:

number of candidates	number of races
14	240
13	54
12	343
11	73
10	35
9	34
8	14
7	10
6	6
5	1