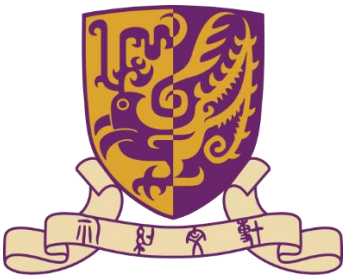

香港中文大學

計算機科學及工程學系

Department of Computer Science and Engineering,
The Chinese University of Hong Kong

Bandits Algorithm, Reinforcement Learning, and Horse Racing Result Prediction

---

Authors:

WONG Tin Wang, David

SZE Muk Hei

Supervisor:

Prof. Michael R. LYU

April 20, 2022

The Chinese University of Hong Kong

Abstract

Faculty of Engineering

Department of Computer Science and Engineering

BSc degree in Computer Science

Bandits Algorithm, Reinforcement Learning, and Horse Racing Result Prediction

By WONG Tin Wang, David, SZE Muk Hei

Horse racing is a popular betting entertainment with long history and many researchers from different countries have attempted to work on horse racing prediction. Different approaches, for instance, deep learning, probabilistic programming, machine learning etc., are experimented while the usage of multi-armed bandit algorithm is rarely applied in the field of horse racing. In this report, we created our own approach which is predicting horse racing results using Random Forest and exploring betting strategy with different multi-armed bandit algorithms and other reinforcement learning algorithms. Furthermore, a model selection using bandit algorithm is adopted to enhance the stability of the horse betting strategies. Throughout the project, we focused on the horse racing in Hong Kong and collected data ranged from 1979 to 2020 with more than 26000 records of races. We demonstrated that an acceptable accuracy for horse racing prediction can be achieved; meanwhile, a list of features having crucial impact on horse racing was determined. Additionally, for bandit algorithms, we tried different possibilities of utilizing it on horse betting as well as model selection. In our attempts, we concluded that it could bet on horses with comparatively higher average return and play at least on the same level with public intelligence but still not sufficient to gain profit from horse racing. We have also attempted to explore horse betting strategies with different reinforcement learning algorithms and concluded that the results are mostly similar to that of the bandit algorithms.

Acknowledgment

Primarily, we would like to thank our supervisor Professor Michael R. Lyu and our advisor Mr. Jen-tse Huang who guided us in doing this final year project as well as provided us with invaluable advice.

Besides, we would like to express our special thanks of gratitude to all students working on their final year project related to horse racing and betting strategies which gave us a good reference to this project.

Table of Content

1. Introduction

1.1 Overview

The focus of this final year project is trying to make horse racing result prediction using random forest model, as well as utilize reinforcement learning and multi-armed bandit algorithms on learning to bet on horse racing. The work done during the first semester will be illustrated throughout this report. This chapter provides a brief introduction to the topic, background information and objectives.

1.2 Motivation

Machine learning is one of the most important fields in artificial intelligence and is growing rapidly. The reason behind this is that machine learning can perform more efficiently than human beings with high accuracy, for instance, the accuracy of diagnosing diabetes reaches 95% using the Naive Bayes algorithm [21]. Working with data, which is one of the major subfields of machine learning across statistics and computer science, has been widely used or conducted in different fields. Time for analyzing the huge amount of data with complex structure has been reduced, which results that many types of research, for example, analyzing data of climate, environment, soil, etc., are conducted to look for optimal parameters and maximize the crop production [41]. With the assistance of machine learning, analyzing complex data such as horse racing data becomes simpler, which arouses our interest in this topic.

Horse racing is one the most famous betting entertainments in Hong Kong which involves a lot of different variables and settings. Data analysis in horse racing is a suitable field for applying machine learning since the amount of data is huge and the structure of horse racing data is complicated for humans while machines can complete the task effectively. With the high transparency of the data of the horses, races courses provided by the Hong Kong Jockey Club, and the attractive gambling profits, motivated us to work on the horse racing result prediction.

Previously, a variety of approaches were used to predict the horse racing result worldwide. For instance, a study has found that the accuracy of using a svm-based committee machine to predict the first-placed horse of a horse racing game in Hong Kong has reached 70.86% with a stable return rate of 8000% by using the static wager strategy for 3 years [58]. Another study shows that over 41% correct bets ratios for the Win bet and more than 36% for the Quinella bet with the

assistant of Linear Discriminant Analysis and Neural Networks in the Poland horse racing can be achieved [10]. Other approaches such as deep probabilistic programming [59], deep learning [37], TensorFlow [15], and reinforcement learning (DQN) [16], were used in horse racing prediction in the previous final year projects. Instead of adopting another approach, reinforcement learning is the focus throughout this project and multi-armed bandit algorithms will be the learning algorithm for the learning process.

1.3 Background

1.3.1 Hong Kong Jockey Club (HKJC)

The Hong Kong Jockey Club is founded in 1884 which is a non-profit organization. It is granted a legal monopoly over all betting entertainments including horse racing, soccer, and mark six. Being the greatest taxpayer and largest community benefactor in Hong Kong [65], the Hong Kong Jockey Club paid $24.4 billion for tax and donated $4.5 billion to the community in 2020. Simultaneously, more than 20000 job opportunities including full-time and part-time jobs [32].

1.3.2 Horse Racing in Hong Kong

The first horse racing began in Happy Valley in 1846 and it turned professional in 1971. Nowadays, horse racing is one of the most popular betting entertainments for local people. In 2020, the amount of betting on horse racing reached 136.1 billion Hong Kong dollars which provided a total revenue of 18.8 billion Hong Kong dollars to the HKJC.

The race season starts in September and ends in July and there are 88 racing fixtures in the 2021/2022 season. Local horse racing events are held in Sha Tin or Happy Valley Racecourse with a total of 3 different tracks. Horse races are classified into different distances including short, middle, and long distance.

Figure 1.1 Views of Racetracks [29]

The rating system of horses classifies each horse into different race classes and there are only horses from the same race class in each race. A new horse can be rated as race class 3 at most depending on their performances in the foreign races and birthplace. At least 5 points will be added to the winning horse and points will be added or deducted from the other horses depending on their final places.

Table 1.1 Rating for Race classes [57]

Rating upper bound	Race classes
120	1
100	2
80	3
60	4
40	5

Handicapping policy is implemented to equalize the horses in the same race. Horses are ordered by their rating in descending order in each race. Additional weights ranged from 113lbs to 133lbs including the weight of their jockey are added to each horse. The horse with the highest rating in each race is required to carry more weights but no more than 133lbs and vice versa. Under some circumstances, for instance, horses aged under 2 or female horses would receive a weight allowance of 2lbs to 5lbs. The freelance jockeys and apprentices would have a weight allowance which is inversely proportional to their winning records [31].

Table 1.2 Weight Allowance for Different Jockeys

	Allowance for Freelance jockeys	Allowance for Apprentice
Less than 20 ridden winners	10lbs	10lbs
Less than 45 ridden winners	7lbs	7lbs
Less than 70 ridden winners	5lbs	5lbs
Less than 95 ridden winners	3lbs	3lbs
Less than 250 ridden winners	2lbs	2lbs
More than 250 ridden winners	0lb	2lbs

The draw of each horse refers to the starting gate where the starting position of that horse is. The draws with smaller numbers are closer to the inner rail and are assigned to each horse randomly before the race. Horses and jockeys would benefit from the draw if the position of draws matches their characteristics.

1.3.3 Betting Rules

The pari-mutuel local pools are adopted by the HKJC and thus, the dividend is shared by the number of winning combinations of a particular pool. It results in different extends of fluctuations of the betting odds according to the amount of bet which reflects the popularity of each horse [56].

Currently, there are 3 distinct betting categories for horse racing which are single-race, multiple-races, and fixed-odds bets.

Table 1.3a Betting Options for Single Race

Single-race bets	Dividend Qualification
Win	Picked the first placed horse
Place	Picked any one of the first three placed horses
Quinella	Picked the first and second placed horses in any order
Quinella Place	Picked any two of the first three placed horses in any order
3 Pick 1 (Composite Win) *	Picked composite containing the first placed horse
Forecast	Picked the first and second placed horses in correct order
Trio	Picked the first three placed horses in any order
Tierce	Picked the first three placed horses in correct order
First 4	Picked the first four placed horses in any order
Quartet	Picked the first four placed horses in correct order

* Horses are grouped into three to four composites based on the betting odds, trainers, or regions

Table 1.3b Betting Options for Multiple Races

Multi-race bets	Dividend Qualification
Double	Picked the first placed horse in each of the two nominated races
Treble	Picked the first placed horse in each of the three nominated races
Double Trio	Picked the first three placed horses in any order in each of the two nominated races
Triple Trio	Picked the first three placed horses in any order in each of the three nominated races
Six Up	Picked the first two placed horses in each of the six nominated races

Table 1.3c Betting Option with Fixed Odds

Fixed-odds bets	Dividend Qualification
Jockey Challenge	Picked the best performing jockey in a race meeting

1.4 Objectives

The objective of this project is to build a model to predict the horse racing results and apply multi-armed bandit algorithms in exploring betting strategy with the horse racing result predictions. Since the data of horses and races are complex and structured comprehensively, distinctive features of horses, racetracks, environment are expected to be classified by their extent of influent to the result of prediction, as well as combined to form an optimal group of parameters in order to make a more accurate prediction of horse racing. In terms of the profits, although applying both static and dynamic wager strategies accompanied with the prediction by an SVM-based committee machine are studied to have 8,000% and 800,000% return rate in a 3-year period [58], the stability of gaining profit under all circumstances would be the prior objective instead of the amount of profit.

In this semester, we focus on improving the performance of the finishing time prediction and exploring different ways to interpret the time prediction model. On the other hand, for the horse betting strategy's part, other reinforcement learning algorithms, for instance, Deep Q Network, Proximal Policy Optimization, Augmented Random Search and Cross Entropy Method are adopted in exploring new betting strategies and evaluating the performance of the multi-armed bandit algorithms in the field of horse betting. In addition, a new approach of model selection using bandit algorithm would be discussed which aims to improve the stability of the horse betting strategies.

Due to the high complexity of distinct betting rules over the world and a wide variety of betting options, this project would only focus on the horse races in Hong Kong only and the explored betting strategies would only be applicable to WIN or PLACE betting options.

2. Preliminary in Horse Racing Prediction

2.1 Overview

In order to utilize and maximize the benefits of the multi-armed bandit algorithms, our approach would be first to create a model which could predict the winner of each race and apply different multi-armed bandit algorithms to find the best betting strategies using the predicted results. In this chapter, the motivation of using this approach would be discussed and the overall procedure of this project would be introduced. Besides, related work of horse racing prediction would be stated and the reason of choosing random forest would be addressed. Furthermore, descriptions of different type of multi-armed bandit algorithms would be included at the end of the chapter.

2.2 Motivation

Previously, there existed numerous approaches from different countries which are targeted at having an accurate prediction in horse racing as well as generate stable profit under all circumstances. For instance, deep learning [12, 29], probabilistic programming [59], machine learning [16], etc., have been studied and applies in the field of horse racing prediction. In contrast, as far as we know, multi-armed bandit algorithm has never been applied to any horse racing related predictions. Thus, in this project, we would like to explore the practicability of adopting multi-armed bandit algorithms in horse racing. Meanwhile, we would like to employ our own approach which is first predict the horse racing result with machine learning and investigate the betting strategy with multi-armed bandit algorithm. The reason of using machine learning for horse racing prediction instead of multi-armed bandit is that there is easier and straight forward approach for time prediction which has been studied previous [10]. Additionally, the accuracy of horse racing prediction with machine learning is acceptable based on the previous projects mentioned, so as to provide stable and reliable datasets for the training with multi-armed bandit algorithms. The reasons of choosing random forest and multi-armed bandit for exploring betting strategies would be discussed in chapter 4 and 5 respectively.

2.3 Overall Procedure

In this project, there are 2 major parts which are horse racing prediction (Random Forest) and betting strategy training (Multi-armed Bandit Algorithm). Before starting the prediction, data collected from different providers would be preprocessed such as encoding, imputing and filtering. Then the processed data would be used in training the prediction model and the intermediate output which includes the predicted place generated by ranking the predicted finishing time would be given out. The intermediate output is then analyzed and used to evaluate the performance of the random forest model. After that, it would be merged with the preprocessed betting odds data and used in the betting strategy learning, and lastly the final output would be returned. The following diagram shows the procedure of this project.

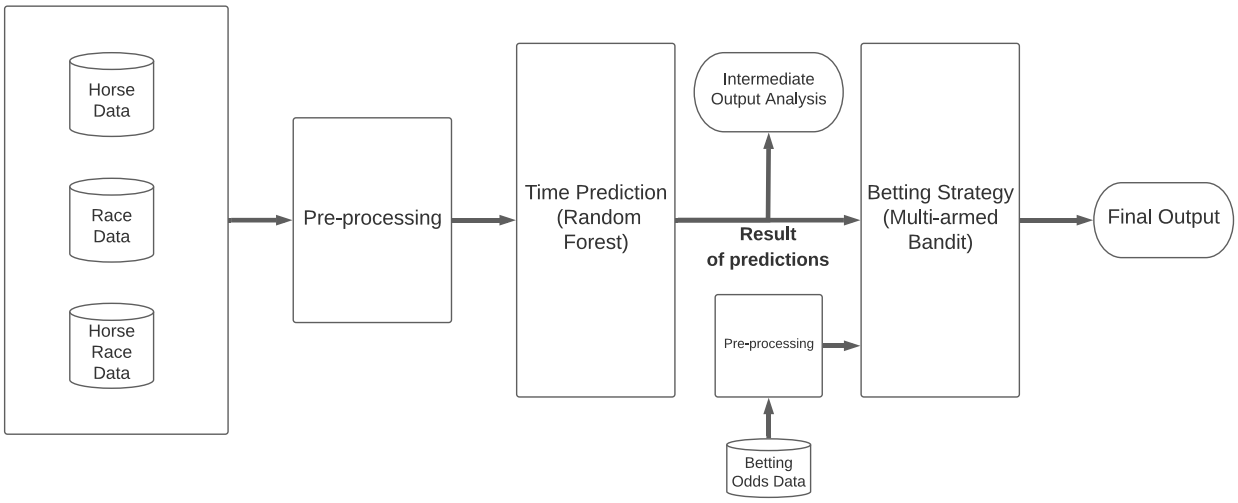

Figure 2.1 Flow Diagram of Overall Procedure

2.4 Time Prediction Techniques

There are alternative ways of doing prediction on the finishing time of horses and there are many previous works that have been done for a similar purpose. Pros and cons of different options would be discussed and the motivation of using a tree-based method would be expressed below with details.

2.4.1 Neural Network

Neural network is a robust model which attempts to simulate functioning as human brain such that it can handle complex combinations of inputs with noises and irrelevant information [38]. Using

neural network in horse racing result prediction is an adequate approach since data of horse racing are complex and the number of features is huge and also complicated nonlinear models can be fitted. It has been extensively studied in LYU1603, LYU1703, LYU1805, as well as Allinson and Merritt also successfully predict the horse racing result using multilayer perceptron in the foreign country which is able to gain 30% to 50% return by betting on the horse with high confidence [47]. However, selecting a suitable combination of hyperparameters such as the number of layers and number of neurons at each layer that is able to avoid overfitting or underfitting takes a lot of testing. Also, it is hard to interpret how detail of the perceptron layers is and model works [38].

2.4.2 Nonlinear Regression Models

Nonlinear regression is a curve fitting approach for analyzing and describing the nonlinear relationship in the data. With the provided training data, nonlinear regression procedures would minimize the residual sum of square (SS) [45]:

$$SS = \sum((y - y')^2) \quad (1)$$

where y is the value of each data and y′ is the value of the selected curve for regression. The process needs to be iterated for a certain number of times and thus, it can adjust the values and improve the result until the improvement is as small as negligible. Nonlinear regression has been studied in agricultural aspect since the interpretability of result is high comparing to neural network and the prediction for unseen data lying out of the range of training data is precise [7]. However, it is possible that the nonlinear regression model converges to local minima or even does not converge after training which may be caused by scattering data or inappropriate selection of equation [45]. Since the finishing time of the horse races are quite scattered and a lot of testing is required for the correct selection of equation, nonlinear regression models are not considered as the most suitable options.

2.4.3 Tree-based Methods

Decision trees are extensively used in machine learning and statistics as a predictive model which summarizes the attributes of the observations given in an understandable and actionable way and predicts the corresponding targeted values of unseen observations [24]. A decision tree consists of a root, node splits and terminal nodes. Root and each node split to separate the data into different branches by classifying the attribute of the data with rules. The terminal nodes consist of a class label which is the targeted value of the data. The decision tree can be easily interpreted by printing out the structure of the tree or the tree path of the testing data which favors us for analyzing the factors that influence the prediction.

Tree ensemble models such as random forest, AdaBoost (Adaptive Boost) and XGBoost (Extreme Gradient Boost) can be other options for tree-based methods. These models combining decision trees tend to have lower bias and variance and avoid overfitting or underfitting, and thus, give a more accurate prediction result [24]. Those benefits match our objectives which are making an accurate horse racing prediction and figuring out which factors have significant influences on horse racing prediction. Since using XGBoost on horse racing prediction is studied in LYU2003 and random forest on horse racing prediction would be our approach to study horse racing prediction.

2.5 Horse Betting Strategies

2.5.1 Bandit Algorithms

In this semester, the bandit algorithms that we will use include Linear upper-confidence-bound (LinUCB) [36], Linear Thompson sampling (LinTS) [2], Contextual Combinatorial MAB (CC-MAB) [14], Neural Bandit [3], Neural upper-confidence-bound (NeuralUCB), and Exponential weights for Exploration and Exploitation (EXP3). We will introduce them in this chapter.

2.5.1.1 Upper-confidence-bound (UCB)

An important problem in picking the best action is to estimate the reward of all actions and play action with highest expected reward. We can estimate true mean reward μ of an action i by taking mean rewards $\hat{\mu}$ of playing this arm in previous t rounds. However, this can differ from the actual reward by Δ. So basically $\hat{\mu} - \Delta \leq \mu \leq \hat{\mu} + \Delta$. And true mean can lie within this interval for certain probability p. UCB algorithms are based on the idea so called "optimism in the face of uncertainty". It picks the one with highest upper confidence bound $\hat{\mu} + \Delta$ because it is possible that the reward can be that high. But how do we set Δ? One example is the UCB1 algorithm.

Deterministic policy: UCB1.
Initialization: Play each machine once.
Loop:

- Play machine j that maximizes $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$, where $\bar{x}_j$ is the average reward obtained from machine j, n_j is the number of times machine j has been played so far, and n is the overall number of plays done so far.

Figure 2.2 UCB1 [9]

In UCB1 algorithm, Δ is set to $\sqrt{\frac{2 \ln n}{n_j}}$, where n is total number of rounds played and n_j is number

of times action j is played. So as more rounds this arm is played, Δ will be smaller and there will be tighter confidence bound and the probability p that the true mean falls in this bound will also be higher which can be proved by Chernoff-Hoeffding inequality, but we won't include the details here. So intuitively, the actions that are less chosen will have a large upper confidence bound, and the algorithm will tend to choose them, and this is one way to explore. After exploring enough, the algorithm will exploit the actions that have higher mean rewards as Δ is small for every action. This represents one way of solving explore-exploit dilemma in stochastic bandit setting.

Standard UCB algorithm doesn't consider the situation where there is context (such as information about an article in article recommendation), which could be related to the reward (whether the article will be clicked). In our project, we will use LinUCB, which is a variant of UCB to handle contextual bandit setting.

Algorithm 1 LinUCB with disjoint linear models.

Inputs: $\alpha \in \mathbb{R}_+$
for $t = 1, 2, 3, \ldots, T$ **do**
  Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
    $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a}}$
  end for
  Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
end for

Figure 2.3 Disjoint LinUCB [36]

In LinUCB, it assumes linear relation of context with the expected reward. That is, $\mathrm{E}[r_{t,a}\,|x_{t,a}] = {x_{t,a}}^T\theta_a^*$, where a is an action, $r_{t,a}\,/x_{t,a}$ is the corresponding reward/context at time t. And there exists a coefficient θ_a^* specific to action a_t that fulfills this relation between $r_{t,a}$ and $x_{t,a}$. In the paper, ridge regression is applied to estimate θ_a^*. And the estimate $\hat{\theta}_a$ will be $({D_a}^T D_a + I_d)^{-1}{D_a}^T c_a$, where D_a is a m x d matrix with rows being d-dimensional context of m seen training samples for

action a at time t, c_a is column vector with m row of corresponding rewards, I_d is d x d identity matrix.

For Δ, it is set to $\alpha\sqrt{x_{t,a}{}^T A_a{}^{-1} x_{t,a}}$, where $A_a = D_a{}^T D_a + I_d$, $\alpha = 1 + \sqrt{\frac{\ln(2/\delta)}{2}}$, δ is hyperparameter to set. And the expected reward will be within the confidence interval with probability at least $1 - \delta$. According to the paper, $\sqrt{x_{t,a}{}^T A_a{}^{-1} x_{t,a}}$ will also reduce as there are more samples. So overall the algorithm will pick action such that upper confidence bound $\mathrm{E}[r_{t,a} | x_{t,a}] + \Delta = x_{t,a}{}^T \hat{\theta}_a + \alpha\sqrt{x_{t,a}{}^T A_a{}^{-1} x_{t,a}}$ is maximized. In our problem, we just set δ to be 0.3.

Algorithm 2 LinUCB with hybrid linear models.

Inputs: $\alpha \in \mathbb{R}_+$
$\mathbf{A}_0 \leftarrow \mathbf{I}_k$ (k-dimensional identity matrix)
$\mathbf{b}_0 \leftarrow \mathbf{0}_k$ (k-dimensional zero vector)
for $t = 1, 2, 3, \ldots, T$ **do**
  Observe features of all arms $a \in \mathcal{A}_t$: $(\mathbf{z}_{t,a}, \mathbf{x}_{t,a}) \in \mathbb{R}^{k+d}$
  $\hat{\boldsymbol{\beta}} \leftarrow \mathbf{A}_0^{-1}\mathbf{b}_0$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{B}_a \leftarrow \mathbf{0}_{d\times k}$ (d-by-k zero matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\left(\mathbf{b}_a - \mathbf{B}_a\hat{\boldsymbol{\beta}}\right)$
    $s_{t,a} \leftarrow \mathbf{z}_{t,a}^\top \mathbf{A}_0^{-1}\mathbf{z}_{t,a} - 2\mathbf{z}_{t,a}^\top \mathbf{A}_0^{-1}\mathbf{B}_a^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a} + \mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a} + \mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{B}_a\mathbf{A}_0^{-1}\mathbf{B}_a^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a}$
    $p_{t,a} \leftarrow \mathbf{z}_{t,a}^\top\hat{\boldsymbol{\beta}} + \mathbf{x}_{t,a}^\top\hat{\boldsymbol{\theta}}_a + \alpha\sqrt{s_{t,a}}$
  end for
  Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
  $\mathbf{A}_0 \leftarrow \mathbf{A}_0 + \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{B}_{a_t}$
  $\mathbf{b}_0 \leftarrow \mathbf{b}_0 + \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{b}_{a_t}$
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
  $\mathbf{B}_{a_t} \leftarrow \mathbf{B}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{z}_{t,a_t}^\top$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
  $\mathbf{A}_0 \leftarrow \mathbf{A}_0 + \mathbf{z}_{t,a_t}\mathbf{z}_{t,a_t}^\top - \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{B}_{a_t}$
  $\mathbf{b}_0 \leftarrow \mathbf{b}_0 + r_t\mathbf{z}_{t,a_t} - \mathbf{B}_{a_t}^\top \mathbf{A}_{a_t}^{-1}\mathbf{b}_{a_t}$
end for

Figure 2.4 Hybrid LinUCB [36]

For the above disjoint LinUCB, there is only one parameter θ_a^* which specific to action a. There is also a variant in the same paper, called hybrid LinUCB, which has a parameter β^* which is shared by all actions. This can potentially be useful so we will use this instead. For hybrid LinUCB, $\mathrm{E}[r_{t,a} | x_{t,a}]$would be $z_{t,a}{}^T\beta_a^* + x_{t,a}{}^T\theta_a^*$, where $z_{t,a}$ is combination of action and

global features, it is article and user in the paper. In our case, we don't have any global features, so we just include features of actions(horses).

2.5.1.2 Thompson Sampling

Different from UCB that estimates the mean reward and confidence interval, Thompson sampling assumes the reward of an action to follow certain probability distribution such as Bernoulli and Gaussian and estimate the parameters for that distribution like mean and variance. A random value is drawn from the distribution to be the estimated reward. As we have better estimate for these parameters when more samples come in, the drawn reward value would have a higher chance of being near to actual mean.

We can take Bernoulli Thompson sampling as an easier example. It first assumes the reward is drawn from Bernoulli distribution, which is parametrized by probability q of getting 1. We should play the actions with highest q to get best estimated reward. To learn about probability of q, Thompson sampling algorithms use Bayes rule.

$$p(q|reward) = \frac{p(reward|q)p(q)}{p(reward)} \propto p(reward|q)p(q) = Bernoulli(q)p(q) \qquad (2)$$

To calculate $p(q|reward)$, we have to know prior $p(q)$. In Bernoulli Thompson sampling, prior is usually chosen to be beta distribution which is parametrized with success rate α and failure rate β. Using beta distribution as the prior makes it easier to calculate $p(q|reward)$ as the calculated result would also be a beta distribution, which is called conjugacy property. Initially $p(q)$ is $Beta(\alpha, \beta)$. After observing the reward, if reward = 0, failure rate β is incremented by 1, $p(q|reward)$ is easily calculated as $Beta(\alpha, \beta + 1)$, if reward = 1, success rate α is incremented by 1, $p(q|reward)$ is calculated as $Beta(\alpha + 1, \beta)$.

The standard procedure for Thompson Sampling is for each arm initialize α, β to be 1, then $Beta(\alpha, \beta)$ will be a uniform distribution in [0,1] range. Then we randomly draw q from $Beta(\alpha, \beta)$. We pick the arm with highest q, observe its reward and update α, β. And continue doing so will eventually get $Beta(\alpha, \beta)$ to be more concentrated around the actual q. The following figure illustrates this idea.

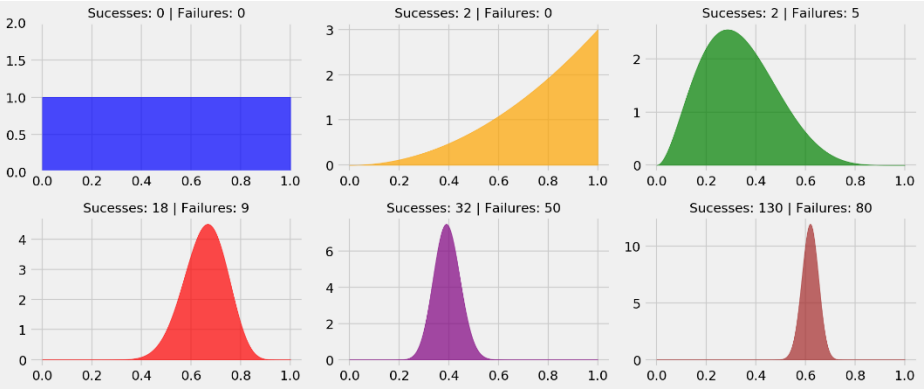

Figure 2.5 Update of beta prior [43]

So, at start, the distribution is more widely spread, and the drawn value is more random. For each action, it's possible to get high reward, making it to be chosen. This achieves exploration. When there are more samples, we get more accurate around the actual mean, and only those action with higher mean will be chosen. This achieves exploitation. It represents a different method from UCB to solve stochastic bandit problem.

Algorithm 1 Thompson Sampling for Contextual bandits

Set $B = I_d, \hat{\mu} = 0_d, f = 0_d$.
for all $t = 1, 2, \ldots,$ **do**
  Sample $\tilde{\mu}(t)$ from distribution $\mathcal{N}(\hat{\mu}, v^2 B^{-1})$.
  Play arm $a(t) := \arg\max_i b_i(t)^T \tilde{\mu}(t)$, and observe reward r_t.
  Update $B = B + b_{a(t)}(t) b_{a(t)}(t)^T$, $f = f + b_{a(t)}(t) r_t$, $\hat{\mu} = B^{-1} f$.
end for

Figure 2.6 LinTS [2]

In our problem, the reward is not simply binary 0 and 1. Therefore, we better use other distributions like Gaussian. Like LinUCB, we would also use a variant of Thompson sampling, which is called Linear Thompson Sampling (LinTS) that work on linear model to support the contextual setting. It relies on Gaussian distribution and uses a Gaussian prior.

To use this algorithm, hyperparameter $v = R\sqrt{\frac{24}{\varepsilon} d \ln \frac{1}{\delta}}$ needs to be set, where d is dimension of context, ε is $\frac{1}{lnT}$ if number of rounds T is known, R is related to the R-sub-Gaussian condition in the paper, which we just find it through experiment and eventually set it to 0.1.

2.5.1.3 Contextual Combinatorial Multi-armed Bandits

Standard bandit algorithms are dealing with the one-armed bandit problem, that is only one action is played at each round. In our horse racing problem, we would like to bet on multiple horses (play multiple actions). This is considered as a combinatorial bandit problem.

Algorithm 2 CC-MAB

Input: $T, h_T, K(t), \mathcal{X}$.
Initialization: context partition $\mathcal{P}_T$; set $C^0(p) = 0, \hat{r}(p) = 0, \forall p \in \mathcal{P}_T$;
for $t = 1, \ldots, T$ **do**:
  Observe arrived arms $\mathcal{M}^t$ and their contexts $\boldsymbol{x}^t = (x_m^t)_{m \in \mathcal{M}^t}$;
  Find $\boldsymbol{p}^t = (p_m^t)_{m \in \mathcal{M}^t}$ such that $x_m^t \in p_m^t, p_m^t \in \mathcal{P}_T, m \in \mathcal{M}^t$;
  Identify under-explored hypercubes $\mathcal{P}^{\text{ue},t}$ and the arm set $\mathcal{M}^{\text{ue},t}$; let $q = |\mathcal{M}^{\text{ue},t}|$;
  if $\mathcal{P}^{\text{ue},t} \neq \emptyset$ **then**: ▷ *Exploration*
    if $q \geq B$ **then** : $\mathcal{S}^t \leftarrow$ randomly pick B arms from $\mathcal{M}^{\text{ue},t}$;
    else: $\mathcal{S}^t \leftarrow$ pick q arms in $\mathcal{M}^{\text{ue},t}$ and other $(B - q)$ as in (6);
  else: $\mathcal{S}^t \leftarrow$ pick B arms as in (7); ▷ *Exploitation*
  for each arm $m \in \mathcal{S}^t$ **do**:
    Observe the quality r_m of arm m;
    Update reward estimation: $\hat{r}(p_m^t) = \frac{\hat{r}(p_m^t)C(p_m^t) + r_m}{C(p_m^t)+1}$;
    Update counters: $C(p_m^t) = C(p_m^t) + 1$;

Figure 2.7 CC-MAB [14]

Say we need to play B actions. The strategy of CC-MAB algorithm is to first play under-explored arms (defined as the number of times the action played is fewer than certain threshold). If number of under-explored arms is less than B, then the remaining arms will be to exploit the arms with highest estimated reward which is simply be the mean.

This algorithm also handles the situation where there is context in a different manner with the two algorithms introduced. It assumes similar context will have similar expected reward. And based on that, it divides the context space into hypercubes of identical size. Such that actions with close context will be grouped into same hypercubes. The estimated reward for these actions will be the mean of rewards of the hypercubes they fall into. And each hypercube will maintain a counter for how many actions belong to this hypercube are played for determining under-explored or not.

The advantage of this algorithm is that it can handle volatile arm set, that is the available arm set in each round is different. Example in our problem is if we define all horses as an action set, there can be some new horses in each round, which we don't have idea on how they perform. If we can group them with some existing horses, it might give us a better idea on that.

To use this algorithm, hyperparameters like threshold K(t), dimension of each hypercube h_T, needs to be defined. We will just follow what's in the paper and set $h_T = T^{\frac{1}{3\alpha+D}}$, $\mathrm{K(t)} = t^{\frac{2\alpha}{3\alpha+D}} \log t$, where T is total number of rounds, t is the current round, D is the context dimension, and α is value satisfying so called Hölder Condition which is related to the assumption of similar context has similar expected reward. We would just set it to be 1 here.

Assumption 1 (Hölder Condition). *There exists* $L > 0$, $\alpha > 0$ *such that for any two contexts* $x, x' \in \mathcal{X}$, *it holds that*

$$|\mu(x) - \mu(x')| \leq L\|x - x'\|^{\alpha} \tag{8}$$

where $\|\cdot\|$ *denotes the Euclidean norm in* $\mathbb{R}^D$.

Figure 2.8 Hölder Condition [14]

2.5.1.4 Neural bandits

In all the aforementioned algorithms, they all have some very restrictive assumptions about the rewards such as linear relation with the context or rewards distribution being Gaussian. This provides easier analysis into theoretical performance but yet may fail for sophisticated rewards that could happen in reality. To alleviate the issue, many has thought of using non-linear or non-parametric approximators. Deep neural network is easily one of them for its strong presentation power. In our project, we attempt to experiment on two neural network-based algorithms, including NeuralBandit [2] and NeuralUCB [62].

Algorithm 1: NeuralBandit1

Data: $\gamma \in [0, 0.5]$ et $\lambda \in]0, 1]$
begin
  Initialize $W_1 \in]-0.5, 0.5[^{N \times K}$
  for $t = 1, 2, ..., T$ **do**
    Context x_t is revealed
    $\hat{k}_t = \arg\max_{k \in [K]} \mathbf{N}_t^k(x_t)$
    $\forall k \in [K]$ on a $\mathbf{P}_t(k) = (1 - \gamma)\mathbf{1}[k = \hat{k}_t] + \frac{\gamma}{K}$
    $\tilde{k}_t$ is drawn from $\mathbf{P}_t$
    $\tilde{k}_t$ is predicted and $y_{t,\tilde{k}_t}$ is revealed
    Compute $\tilde{\Delta}_t$ such as $\tilde{\Delta}_t^{n,k} = \frac{\lambda \hat{s}_t^{n,k} \delta_t^{n,k} \mathbf{1}[k_t = k]}{\mathbf{P}_t(k)}$
    $W_{t+1} = W_t + \tilde{\Delta}_t$

Figure 2.8 Neural Bandit [3]

NeuralBandit uses a committee of K neural networks, where each is for computing the reward for each arm out of K arms. It uses epsilon greedy for exploration strategy, which is simply doing exploration for a fixed probability. The parameters of neural networks are updated by gradient descent weighted by the probability of arm being chosen. Overall, it has a simple design choice, which could make it robust but might not be efficient especially on exploration. In our experiment, is set to 0.1. The neural network is chosen to be with one hidden layer with sigmoid activation, which coincides with original paper. Hidden layer size is set to be 48.

Algorithm 1 NeuralUCB

Input: Number of rounds T, regularization parameter λ, exploration parameter ν, confidence parameter δ, norm parameter S, step size η, number of gradient descent steps J, network width m, network depth L.
Initialization: Randomly initialize $\boldsymbol{\theta}_0$ as described in the text
Initialize $\mathbf{Z}_0 = \lambda \mathbf{I}$
for $t = 1, \ldots, T$ **do**
  Observe $\{\mathbf{x}_{t,a}\}_{a=1}^{K}$
  for $a = 1, \ldots, K$ **do**
    Compute $U_{t,a} = f(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1}) + \gamma_{t-1}\sqrt{\mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})^\top \mathbf{Z}_{t-1}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})/m}$
    Let $a_t = \operatorname{argmax}_{a \in [K]} U_{t,a}$
  end for
  Play a_t and observe reward r_{t,a_t}
  Compute $\mathbf{Z}_t = \mathbf{Z}_{t-1} + \mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})\mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})^\top / m$
  Let $\boldsymbol{\theta}_t = \text{TrainNN}(\lambda, \eta, J, m, \{\mathbf{x}_{i,a_i}\}_{i=1}^{t}, \{r_{i,a_i}\}_{i=1}^{t}, \boldsymbol{\theta}_0)$
  Compute

$$\gamma_t = \sqrt{1 + C_1 m^{-1/6}\sqrt{\log m} L^4 t^{7/6} \lambda^{-7/6}} \cdot \left(\nu\sqrt{\log \frac{\det \mathbf{Z}_t}{\det \lambda \mathbf{I}} + C_2 m^{-1/6}\sqrt{\log m} L^4 t^{5/3} \lambda^{-1/6} - 2\log\delta} + \sqrt{\lambda} S\right)$$
$$+ (\lambda + C_3 t L)\left[(1 - \eta m \lambda)^{J/2}\sqrt{t/\lambda} + m^{-1/6}\sqrt{\log m} L^{7/2} t^{5/3} \lambda^{-5/3}(1 + \sqrt{t/\lambda})\right].$$

end for

Figure 2.9 NeuralUCB [62]

While for NeuralUCB, it uses just one neural network for all arms and manages to introduce UCB for exploration which could be more efficient. Unlike LinUCB, the definition of term can be seen to be very complicated due to complexity of deep neural network. Therefore, we don't expand the explanation here like what was did above. In our experiment, regularization parameter λ is set to 1, exploration υ is set to 0.0001, confidence parameter δ is set to 0.00005.

2.5.1.5 Adversarial bandit

All the aforementioned bandit algorithms belong to the class of stochastic bandit algorithms, which assumes the rewards follow certain fixed distributions. For adversarial bandit algorithms, it doesn't assume any reward to follow certain distribution but rather determined by an adversary that could even possibly setting rewards against actions we would take. It can be better to handle uncertainty such as non-stationary reward where reward distribution is ever changing. This makes it a good candidate for model selection problem, in which the involved models may behave differently as time goes on. Therefore, we will leverage this class of algorithms to help us compare different reinforcement learning algorithms.

Algorithm Exp3
Parameters: Real $\gamma \in (0, 1]$
Initialization: $w_i(1) = 1$ for $i = 1, \ldots, K$.

For each $t = 1, 2, \ldots$

1. Set
$$p_i(t) = (1 - \gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K} \qquad i = 1, \ldots, K.$$
2. Draw i_t randomly accordingly to the probabilities $p_1(t), \ldots, p_K(t)$.
3. Receive reward $x_{i_t}(t) \in [0, 1]$.
4. For $j = 1, \ldots, K$ set
$$\begin{aligned} \hat{x}_j(t) &= \begin{cases} x_j(t)/p_j(t) & \text{if } j = i_t \\ 0 & \text{otherwise,} \end{cases} \\ w_j(t+1) &= w_j(t) \exp\left(\gamma \hat{x}_j(t)/K\right) . \end{aligned}$$

Figure 2.10 EXP3 [8]

One classic algorithm for solving the adversarial bandit problem is called EXP3 (**Exp**onential weights for **Exp**loration and **Exp**loitation). It maintains a list of weights for each action. It would select actions with the largest weight with probability γ and randomly select actions for $1 - \gamma$. The weights are updated by multiplying the current weight by the exponential of the estimated reward for the current round. This makes it responsive to reward changes. In our experiment, γ is set to be 0.1.

2.5.2 Deep Q Network

Deep Q-Network (DQN) is a deep reinforcement learning algorithm which has been demonstrated by DeepMind that it can be applied to solve a wide range of Atari games [46]. Deep Q Learning with MLP policy has been applied to horse betting based on WIN betting last year and achieved an acceptable result by [16]. In this semester, we will try to apply Deep Q Learning in horse betting based on PLACE betting and compare the performance with other reinforcement learning algorithms.

2.5.2.1 Q Learning

Q Learning is a model-free reinforcement learning algorithm which allows agents learning to act optimally in Markovian domains by experiencing the results of actions, without requiring them to build maps of the domains [46]. Q Learning is similar to the temporal difference method which concerns the rewards or penalties that are not immediately observable in the environment in order

to estimate the value of the current state. Q Learning is based on Q function which is a state-value action function on a policy π defined as follow:

$$Q^{\pi}(s, a) \tag{3}$$

where s is that state and a is the action, and the output of the Q function is the expected reward of taking action a at state s following the policy π. In addition, the Q function with an optimal policy which based on Bellman optimality equation is defined as follow:

$$Q^{*}(s, a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^{*}(s', a')\right] \tag{4}$$

where r is the immediate reward and gamma is the discount factor, s' and a' are the next state and action taken in the next state respectively. This expected reward of the current state is computed by both immediate reward r and the maximum reward can be achieved by taking next action a' in state s'. For the discount factor γ $(0 < \gamma < 1)$ reduces the reward in the farther states since the selected action a in the current state s has less influence on them.

For Q Learning algorithm, an approach of iterative update is performed using the Q function in order to estimate the Q value and reduce the Bellman error which is defined as follow [46]:

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$
    $s \leftarrow s'$;
  until s is terminal

Figure 2.11 Q Learning [46]

where α is the learning rate deciding the degree of steps that the agent takes in order to achieve convergence in the training.

Although Q Learning has been proved to achieve convergence, it is not suitable to be applied to horse betting since the number of states in horse betting environment is large and the details of most of the states are different from each other, for instances, the odds of horses or the condition of the races. If Q Learning is used, it is difficult for the agent to obtain an optimal horse betting strategy since there is lack of generalization. When there are new races and agent would simply treat them as new states since the new races are different from the previous states and agent is not able to find the corresponding Q value from the previous results, and thus, the result of training

might be hindered, and Deep Q Network would be alternative option that applied to horse betting instead of Q Learning.

2.5.2.2 Deep Q Learning

Deep Q Learning is a proposed by DeepMind which combined reinforcement learning and deep neural network in order to solve the challenge mentioned in the previous section [46]. A Q network which is a neural function approximator with weight θ to estimate the action-value function $Q(s, a;\ \theta)\ \approx\ Q^*(s, a)$, and is trained by the following loss function in each iterate i.

$$L_i(\theta_i)\ =\ \mathbb{E}_{s,a,r,s' \sim \rho(.)}\left[\left(y_i\ -\ Q(s, a;\ \theta_i)\right)^2\right] \tag{5}$$

where $y_i = r + \gamma \max_{a'} Q(s', a';\ \theta_{i-1})$ and ρ is the distribution of transition $\{s, a, r, s'\}$ collected from the environment.

In contrast, this leads to another problem which is the moving target problem since the target depends on the network weight and it would potentially lead to a harmful correlation and increase the difficulty of training the network. Thus, to solve this problem, a copy of neural network called target network would be kept providing a fixed target for the main Q network. The target network would not be trained in every step, instead it would be synchronized with the parameters of the main Q network every long enough interval, and thus, the target network could enhance the convergence of the Q network and reduce the difficulty of training a moving target network.

In addition, Q Learning is an off-policy reinforcement learning algorithm which adopts the greedy strategy $a\ =\ \max_a Q(s, a;\ \theta)$, and may lead to problem of lacking exploration. To solve this problem, epsilon-greedy strategy would be used for selecting the action which states that there will be probability of epsilon selecting an action following the greedy strategy and selecting an action randomly with probability 1 - epsilon.

Experience Replay is another technique used by DeepMind in order to enhance the stability of updating the neural network by storing experience of the agent $e_t = (s_t, a_t, r_t, s_{t+1})$ at every timestamp t in the replay buffer D with a capacity N, and the old experience would be overwritten when the number of experiences exceeds the capacity N. Instead of using the latest transition for computing the loss and the gradient, a random minibatch of transitions would be sampled and used for computing which enhance the efficiency of reusing the old transitions in each update and break the strong correlation between samples and reduce the variance of update. The following the Deep Q Learning algorithm designed by DeepMind.

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Figure 2.12 Deep Q Learning [46]

2.5.3 Proximal Policy Gradient

2.5.3.1 Policy Gradient

Reinforcement learning's goal is to learn a policy π that maximizes expected total reward $E_{\tau\sim p_\pi(\tau)}[R(\tau)] = E_{\tau\sim p_\pi(\tau)}[\sum_{t=1}^{T} r(s_t, a_t)]$ over trajectories $\tau = \{s_1, a_1, s_2, a_2, \ldots, s_T, a_T\}$. Say the policy is a model parameterized by θ, we would want to update θ to maximize the expected reward $\bar{R}_\theta = E_{\tau\sim p_\theta(\tau)}[R(\tau)] = \sum_\tau R(\tau) p_\theta(\tau)$. Policy gradient updates θ using gradient ascent: $\nabla\bar{R}_\theta = \sum_\tau R(\tau)\nabla p_\theta(\tau)$. And in practice, we estimate expected reward by mean reward of sampling N games $\frac{1}{N}\sum_{n=1}^{N} R(\tau^n)$ and the gradient is formulated as $\frac{1}{N}\sum_{n=1}^{N} R(\tau^n)\nabla\log(p_\theta(\tau))$.

2.5.3.2 PPO

The problem of policy gradient is that after each time it updates θ, we have to let the policy interact with the environment and sample the data again as the previous sampled data is from a different θ which gives a different expected reward as the probabilities of choosing actions are different. And it is very inefficient. This is called on-policy which means the policy to be updated is the same as the policy that interacts with the environment. Instead, we would want to do off-policy where two policies don't have to be the same and we can simply use the data sampled by another policy to update the policy. To achieve this, we can use importance sampling which approximates $E_{x\sim p}[f(x)]$, the expected value of function $f(x)$ from distribution p, with $E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]$ where q is another distribution. With importance sampling, the expected reward for policy with parameter θ is $E_{\tau\sim p_\theta(\tau)}[R(\tau)]$ can be approximated by $E_{\tau\sim p_{\theta'}(\tau)}\left[R(\tau)\frac{p_{\theta'}(\tau)}{p_\theta(\tau)}\right]$ where θ' is a different parameter with θ. And the gradient becomes $\frac{1}{N}\sum_{n=1}^{N}\frac{p_{\theta'}(\tau^n)}{p_\theta(\tau^n)}R(\tau^n)\nabla\log(p_\theta(\tau))$,

However, as the reward distribution of θ' and θ can have very different variance, causing that we don't get a good estimation of the mean of θ when we don't sample enough time. In PPO, it appends the original objective function $J^{\theta'}(\theta) = E_{\tau \sim p_{\theta'}(\tau)}\left[R(\tau)\frac{p_{\theta'}(\tau)}{p_{\theta}(\tau)}\right]$, the expected reward, with a constraint $-\beta KL(\theta, \theta')$ where $KL(\theta, \theta')$ is the KL divergence between θ and θ'. This makes sure that θ and θ' cannot differ by much and resolves the problem. And the objective function becomes $J_{PPO}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta KL(\theta, \theta')$.

2.5.4 Augmented Random Search

Augmented Random Search is a model-free reinforcement learning algorithm which enhanced the performance of a traditional Basic Random Search algorithm. The performance of solving MuJoCo tasks using Augmented Random Search has been discussed and the computational efficiency of Augmented Random Search is at least 15 times higher than the fastest model-free methods on the benchmarks [42]. In this semester, we would try to apply Augmented Random Search to horse betting strategy learning.

2.5.4.1 Basic Random Search

Random Search is one of the oldest and simplest numerical optimization methods which does not require the computation of gradient of the problem. Random Search uniformly and randomly chooses a direction on the sphere in parameter space, and then optimizes the function along this direction [42]. The follow algorithm is the details of Basic Random Search.

Algorithm 1 Basic Random Search (BRS)

Hyperparameters: step-size α, number of directions sampled per iteration N, standard deviation of the exploration noise ν
Initialize: $\theta_0 = \mathbf{0}$, and $j = 0$.
while ending condition not satisfied **do**
  Sample $\delta_1, \delta_2, \ldots, \delta_N$ of the same size as θ_j, with i.i.d. standard normal entries.
  Collect $2N$ rollouts of horizon H and their corresponding rewards using the policies

$$\pi_{j,k,+}(x) = \pi_{\theta_j + \nu\delta_k}(x) \quad \text{and} \quad \pi_{j,k,-}(x) = \pi_{\theta_j - \nu\delta_k}(x),$$

  with $k \in \{1, 2, \ldots, N\}$.
  Make the update step:

$$\theta_{j+1} = \theta_j + \frac{\alpha}{N}\sum_{k=1}^{N}\left[r(\pi_{j,k,+}) - r(\pi_{j,k,-})\right]\delta_k\,.$$

  $j \leftarrow j + 1$.
end while

Figure 2.13 Basic Random Search [42]

The idea of Basic Random Search is exploring in the parameter space instead of action space and optimizing the reward through optimizing over the policy parameter theta [42]. The parameters

theta would be perturbed by a constant noise v and a random number δ generated with an independent and identically distributed standard normal entries. The reward $r(\theta + v\delta)$ and $r(\theta - v\delta)$ would be collected by applying actions based on $\pi(\theta + v\delta)$ and $\pi(\theta - v\delta)$ respectively and the parameter theta would be updated by the average $\Delta = \frac{1}{N}\sum[r(\theta + v\delta) - r(\theta - v\delta)\,]\,\delta$ for all the δ with the rewards of the perturbed θ and a learning rate α.

2.5.4.2 Enhancements of Basic Random Search

There are some problems exists in the algorithm of Basic Random Search which may potentially influent the performance of the algorithm and are solved in the Augmented Random Search algorithm.

First and the foremost, the variation of the difference between rewards $r(\theta + v\delta)$ and $r(\theta - v\delta)$ can vary significantly when the learning rate α is fixed. For instance, when the average Δ fluctuates rapidly, updating $\theta_{j+1} = \theta_j + \alpha\Delta$ would become a difficult task. If learning rate α is 0.001 and Δ is 1, then $\alpha\Delta$ would be 0.001 while Δ is 1000 and $\alpha\Delta$ would be 1 and this harms the algorithm to converge parameter θ towards the values that maximize the rewards. In order to solve this problem, scaling the rewards collected at each iteration by a standard deviation σ_R would be adopted and thus, the value of step size can be reduced and enhance the stability of the training process [42].

In addition, in order to ensure that equal weight is put on different input states, normalization of the states is required. For instance, comparing 2 different state components C_1 and C_2 where C_1 is in range $[90, 100]$ and C_2 is in range $[-1, 1]$ respectively. If normalization is not applied, C_2 can make no difference to the results when the value of C_2 changed. Suppose $C_1 = 95$ and $C_2 = 1$ and the average of C_1 and C_2 would be $\frac{(95+1)}{2} = 48$, then keep C_1 unchanged and modify the value of C_2 to -1 and the new average is $\frac{(95-1)}{2} = 47$ which shows that even C_2 has decreased from the maximum value to minimum value while the results has no significant difference. If normalization is applied to the state component before computing the results, for instance, the normalized $C_1 = \frac{(95-90)}{(100-90)} = 0.5$ and normalized $C_2 = \frac{(1+1)}{(1+1)} = 1$ respectively and the new average of C_1 and C_2 would be $\frac{(0.5+1)}{2} = 0.75$. When the value of C_2 drops to -1 and the normalized value of updated $C_2 = \frac{(1-1)}{2} = 0$ and the normalized average would be $\frac{(0.5)}{2} = 0.25$ which shows that the influence of the decrement of C_2 becomes more significant and ensures that all state components have an equal influence on the results.

Lastly, instead of computing the average reward in each iteration, Augmented Random Search only uses the top b rewards which aims to maximizing the collected rewards. In order to compute the rewards $r(\theta + v\delta)$ and $r(\theta - v\delta)$ would be collected by applying actions based on $\pi(\theta +$

$v\delta)$ and $\pi(\theta - v\delta)$ respectively, and the average computed would be greatly reduced if there exist some rewards that are relatively smaller than the other rewards which becomes an obstacle to the purpose of maximizing the collected rewards. Augmented Random Search algorithm would sort the rewards in descending order based on the key $max(r(\theta + v\delta), r(\theta - v\delta))$ and select top b rewards for updating step with direction that obtained high rewards in average. The algorithm of Augmented Random Search is shown as below.

Algorithm 2 Augmented Random Search (ARS): four versions **V1**, **V1-t**, **V2** and **V2-t**

1: **Hyperparameters:** step-size α, number of directions sampled per iteration N, standard deviation of the exploration noise ν, number of top-performing directions to use b ($b < N$ is allowed only for **V1-t** and **V2-t**)
2: **Initialize:** $M_0 = \mathbf{0} \in \mathbb{R}^{p \times n}$, $\mu_0 = \mathbf{0} \in \mathbb{R}^n$, and $\Sigma_0 = \mathbf{I}_n \in \mathbb{R}^{n \times n}$, $j = 0$.
3: **while** ending condition not satisfied **do**
4: Sample $\delta_1, \delta_2, \ldots, \delta_N$ in $\mathbb{R}^{p \times n}$ with i.i.d. standard normal entries.
5: Collect $2N$ rollouts of horizon H and their corresponding rewards using the $2N$ policies

$$\textbf{V1:} \begin{cases} \pi_{j,k,+}(x) = (M_j + \nu\delta_k)x \\ \pi_{j,k,-}(x) = (M_j - \nu\delta_k)x \end{cases}$$

$$\textbf{V2:} \begin{cases} \pi_{j,k,+}(x) = (M_j + \nu\delta_k)\,\mathrm{diag}\,(\Sigma_j)^{-1/2}\,(x - \mu_j) \\ \pi_{j,k,-}(x) = (M_j - \nu\delta_k)\,\mathrm{diag}(\Sigma_j)^{-1/2}(x - \mu_j) \end{cases}$$

for $k \in \{1, 2, \ldots, N\}$.
6: Sort the directions δ_k by $\max\{r(\pi_{j,k,+}), r(\pi_{j,k,-})\}$, denote by $\delta_{(k)}$ the k-th largest direction, and by $\pi_{j,(k),+}$ and $\pi_{j,(k),-}$ the corresponding policies.
7: Make the update step:

$$M_{j+1} = M_j + \frac{\alpha}{b\sigma_R} \sum_{k=1}^{b} \left[r(\pi_{j,(k),+}) - r(\pi_{j,(k),-})\right] \delta_{(k)},$$

where σ_R is the standard deviation of the $2b$ rewards used in the update step.
8: **V2** : Set μ_{j+1}, Σ_{j+1} to be the mean and covariance of the $2NH(j+1)$ states encountered from the start of training.[2]
9: $j \leftarrow j + 1$
10: **end while**

Figure 2.13 Augmented Random Search [42]

2.5.5 Cross-entropy Method

The Cross Entropy Method is an evolutionary algorithm which is used for optimizing parameterized policy which has been applied to find optimal strategy of playing Tetris – one of the most popular computer games [33]. The performance of Cross Entropy Method has been proved to outperform other reinforcement learning algorithms at that period by almost 2 orders of magnitude surprisingly. In this semester, we would try to apply Cross Entropy Method to find the optimal horse betting strategy with our finishing time prediction results.

2.5.5.1 Main Idea of Cross-entropy Method

The Cross Entropy Method is a derivative-free optimization techniques which can find the optima without computing derivatives or backpropagation. Instead, Cross Entropy Method is trying to maintain a distribution of possible solutions and update the corresponding distribution at each iteration. In order to use the Cross-Entropy Method to find horse betting strategy with reinforcement learning, we apply the Cross-Entropy Method to optimize the weights of different input features by fitting a distribution to it.

2.5.5.2 Procedures

Suppose $f_t \sim N(\mu_t, \sigma_t^2)$ is the distribution of the parameter vector at iteration t, the Cross Entropy Method starts with sampling n vectors $w_1, \ldots, w_n$ from an initial probability distributions f_0 and collect the corresponding $S(w_1), \ldots, S(w_n)$ where S is the general real-values objective function. Afterward, the top $p\%$ of samples would be selected as elite set where p is the selection ratio and $0 < p < 100$. The selected samples would be used for updating the mean of the distribution by the following equation.

$$\mu_{t+1} := \frac{\sum_{i \in I} w_i}{|I|} \tag{6}$$

and the deviation of the distribution would be updated by the following equation

$$\sigma_{t+1}^2 := \frac{\sum_{i \in I} (w_i - \mu_{t+1})^T (w_i - \mu_{t+1})}{|I|} \tag{7}$$

where $I \subseteq \{1, 2, \ldots, n\}$ is the set of indices of selected samples. Then, the distribution parameters obtained would update the current distribution and the process would be iterated until convergence.

2.5.5.3 Early Convergence Problem

In contrast, an early convergence problem would exist since the speed of distribution concentrating to a single point is fast which limits the applicability of the Cross Entropy Method in reinforcement learning as it always converges to suboptimal policies. Thus, in order to avoid the early convergence problem, noise would be added to the distribution during the updating phase at each iteration t using the following equation.

$$\sigma_{t+1}^2 := \frac{\sum_{i \in I} (w_i - \mu_{t+1})^T (w_i - \mu_{t+1})}{|I|} + Z_{t+1} \tag{8}$$

where Z_{t+1} is the extra noise and a constant vector depending only on t.

3. Data Preparation

3.1 Collection of data

There are several types of historical data provided by the Hong Kong Jockey Club and other third-party providers. All providers are compared by us with respect to the cost and the integrity of data and three of them are chosen to be the data sources of our project.

3.1.1 Data Guru

Data Guru is a third-party provider which has created a database storing formatted data and provided json and csv files for downloading. The source of their data is the Hong Kong Jockey Club which includes the historical data of horses, races, and horse races from 1979 to 2021. The horse-races data is the details of the horse in a particular race including the gear that the horse equipped, the declared weight, the rating, and the final betting odds of the horse, etc. The datasets contain 26568 records of races, 13919 records of horses and 320043 records of a horse race.

3.1.2 The Hong Kong Jockey Club

The Hong Kong Jockey Club is the official data provider for horse racing since 1979. Although all data and records are provided and the difficulties and efficiency to collect the data directly from the Hong Kong Jockey Club is high, partial of data are collected from the Hong Kong Jockey Club website, which is the details of different distance interval of each horse race, to construct the missing features provided from Data Guru. There are in total 127297 records since 2008 are collected from the Hong Kong Jockey Club, and successfully appended to the data files from Data Guru by joining the identifier, which indirectly verified the integrity of the data from Data Guru.

3.1.3 hkHorse

hkHorse is a third-party provider which provides both raw and processed data with a membership system. The source of their data is the Hong Kong Jockey Club, and they have their own analytic system and database to store the data collected from the official website. Since the betting odds fluctuation are not provided by the Hong Kong Jockey Club while only the final betting odd of Win is provided for each horse. The fluctuations of betting odds, including Win and Place, of each horse are collected from hkHorse. There are 43690 records in total since 2019 which includes more than 1500 races.

3.2 Description of Data

3.2.1 Races Data

The races data contains 26568 records which shows the details of a horse race from 1979 to 2021. The following table describes the features of the races data collected from Data Guru.

Table 3.1 Features of a Raca Records

Features	Description	Types	Samples
raceid	A unique id of a race constructed with race date, distance, track type and race season id	Categorical	1979-09-22-001-1235-Grass
raceidseason	Index of a race in race season	Index	001
racedate	Date of the race	Index	1979-09-22
racetrack	Racecourse of the race	Categorical	HV, ST
tracktype	Type of racetrack of the race	Categorical	Turf, Grass, Sand, AWT
course	Width of inner rail and outer rail of the track	Categorical	(Show in later table)
distance	Distance of the race (in meters)	Categorical	1000, 1200, 1400, 1600, 1650, 1800, 2000, 2200, 2400
going	The condition of the track	Categorical	(Show in later table)
raceclass	The race class of horses in the race	Categorical	1, 2, 3, 4, 5

Partial of identical values of the course feature refers to distinct width since the racecourses are different. Table 3.2 describes the details of the course.

The values of the going feature take penetrometer and clegg hammer for Turf Track and All Weather Track (AWT) as references, respectively. In addition, the opinion from jockeys, the surface of the turf, and weather conditions are references for the decision making. Tables 3.3a and 3.3b describe the details of going.

Table 3.2 Details of Different Courses

Course	Widths of track in Sha Tin Racecourse (ST)	Widths of track in Happy Valley (HV)
A	30.5M	30.5M
A+2	28.5M	28.5M
A+3	27.5M	/
B	26.0M	26.5M
B+2	24.0M	24.5M
B+3	/	23.5M
C	21.3M	22.5M
C+3	18.3M	19.5M

Table 3.3a Details of Going (Turf Track)

Going	Description	Penetrometer Reading
F	Firm	< 2.50
G/F	Good To Firm	2.50 - 2.75
G	Good	2.75 - 3.00
G/Y	Good To Yielding	3.00 - 3.25
Y	Yielding	3.25 - 3.50
Y/S	Yielding To Soft	> 3.50
S	Soft	/
H	Heavy	/

Table 3.3b Details of Going (All Weather Track)

Going *	Description
WF	Wet Fast
FT	Fast
GD	Good
SL	Slow
WS	Wet Slow
RA	Rain Affected
NW	Normal Weathering

*Higher the clegg hammer reading, firmer the surface on normal weather condition

3.2.2 Horses Data

The horses data contains 13919 records which shows the details of a horse. The following table describes the features of the horses data collected from Data Guru.

Table 3.4 Features of a Horse Record

Features	Descriptions	Types	Samples
horseid	A unique id of the horse	Categorical	HK_2016_A061
horsename	A unique name of the horse	Categorical	RATTAN
country	The country of the horse	Categorical	NZ, AUS, IRE
colour	The colour of the horse	Categorical	Brown, Grey, Brown/Grey
sex	The sex of the horse	Categorical	Gelding, Mare, Colt
importtype	The import type of the horse	Categorical	PP, PPG, SG
owner	The owner of the horse	Categorical	Zippalanda Syndicate
sire	The sire of the horse	Categorical	Savabeel
dam	The dam of the horse	Categorical	Grand Princess
damsire	The dam's sire of the horse	Categorical	Last Tycoon
url	A url linked to the HKJC data source	/	/
age	The current age of the horse	Real Value	/

3.2.3 Horse Race Data

The horse race data contains 320043 records which shows the details of a horse participating in a race. For instance, an identical horse would be included in different records since it has participated in different races. The table below describes the features of the horse race data collected from Data Guru.

Table 3.5 Features of a Horse Race Record

Features	Descriptions	Types	Samples
horseid	A unique id of the horse	Categorical	HK_2016_A061
raceid	A unique id of the race	Categorical	1979-09-22-001-1235-Grass
place	The final place of the horse	Categorical	1 – 14, DISQ, DNF
draw	The draw where the horse starts racing	Categorical	Integer with range of 1 - 14
rating	The rating of the horse	Real Value	Decimal with range of 1 – 144
trainer	The trainer of the horse	Categorical	R Gibson
jockey	The jockey riding the horse	Categorical	K H Chan
lengthbehindwinner	The distance between the first placed horse and the horse in the race (in horse length)	Real Value	9-½, 6-¼, 5
winodds	The final WIN odds of the horse	Real Value	/
actualweight	The actual weight of the horse	Real Value	Integer with range of 113 – 133
position{1 – 6}	The place of the horse at different distance intervals	Categorical	1 – 14, DISQ, DNF
finishtime	The finishing time of the horse	Real Value	/
declaredweight	The declared weight of the horse	Real Value	/
gear	The gear equipped by the horse	Categorical	B, TT, B/TT

The gear equipment is restricted by the Hong Kong Jockey Club due to the protection of horses and fairness of competition. For instance, horses are required to undergo testing with gear equipped or proved that they participated in a foreign race with that gear equipped before the race starts. The following tables describe the details of each gear [30].

Table 3.6 Details of Gears

Gear	Full Name of Gear	Description
B	Blinkers	Blocking the sight behind the horse
BO	Blinkers with one cowl only	
V	Visor	Same as blinkers but with a small gap
P	Pacifier	Protecting the eyes of horse from sand
PC	Pacifier with cowls	Same as pacifier with blinkers
PS	Pacifier with 1 cowl	
CP	Sheepskin Cheek Pieces	Similar to blinkers but provides a wider sight for the horse
CO	Sheepskin Cheek Piece 1 side	
SR	Shadow Roll	Blocking the sight under the eyes of horse
SB	Sheepskin Browband	Blocking the sight above the eyes of horse
H	Hood	Reduce the sound from the surroundings and movement of the ears of horse
E	Ear Plugs	Reduce the sound from the surroundings
XB	Crossed Nose Band	Preventing the horse from opening their mouth
CC	Cornell Collar	Preventing intermittent dorsal displacement of the soft palate during high-speed exercise
TT	Tongue Tie	Preventing the horse from playing with their tongue

3.2.4 Betting Odds Data

The betting odds data contains 43690 records including both Win and Place betting odds fluctuation with more than 1500 races. The number of betting odds are different depending on the starting time of each race, since the Hong Kong Jockey Club will provide the starting odds at noon the day before the race starts. hkHorse records the betting odds from time to time once the official data is released, and the time intervals varies depending on the how close the starting time of the race is. The following tables describe the fluctuation of betting odds which includes 85 records of betting odds collected from the hkHorse.

Table 3.7 Features of a Betting Odds Record

Features	Descriptions	Types	Samples
horseid	A unique id of the horse	Categorical	HK_2016_A061
raceid	A unique id of the race	Categorical	1979-09-22-001-1235-Grass
oddtype	The type of betting odds	Categorical	W, P
odd_1	The betting odds released by the Hong Kong Jockey Club	Real Value	/
odd_{2 – 12}	The betting odds after the release of betting odds (per hour)	Real Value	/
odd_{13 – 28}	The betting odds between 12 hours after release of betting odds and 2.5 hours before the race starts	Real Value	/
odd_{29 – 55}	The betting odds from 2.5 hours to 30 minutes before the race starts (per 5 minutes)	Real Value	/
odd_{56 – 84}	The betting odds in the last 29 minutes (per minute)	Real Value	/
odd_85	The final betting odd	Real Value	/

The periods that the betting odds fluctuate the most are the beginning of the odd release and the last moment before the race starts. Thus, betting odds within those periods are collected to preserve the fluctuation. The middle period of fluctuation is relatively small since gamblers seldom place bets at midnight or 2.5 hours before the races start. Thus, a length-based interval is used to collect the betting odds within this period such that the more records provided by hkHorse, the larger the time interval is used or vice versa.

3.3 Analysis of Data

In this section, relationships between features would be carefully studied in order to figure out to what extent some factors may influence the prediction of the horse racing result.

3.3.1 Relationship between Performance and Race Classes

The rules of race classes are introduced in section 1.3.2. In general, the race classes that each horse belongs to depends on the rating of the horse and details of the rating system will be discussed in section 3.3.3 which is about the relationship between the rating and the place. There are 5 common race classes which are only for the local horse racing, and they are 1, 2, 3, 4, 5. Furthermore, there are race classes such as G1, G2, Griffin, 4YO etc., which are for the horses with rating exceeds the race class 1 upper bound, unrated horses or horses that are 4 years old at that moment. However, the number of some races of the uncommon race classes are relatively small and those races are only held with 1 or 2 distances which is difficult to illustrate the result with a graph, thus, only the races of the 5 common race classes are discussed below.

The graph below shows the average finishing time of different race classes in races with different distances. There is a slight increase in the average finishing time across the race classes in the race of same distance except for 2000M and 2200M races. The reason for the exceptional case is that the Hong Kong Jockey Club has seldom organized the 2000M and 2200M races for race classes 1, 2 and 3, as well as the number of races in the exceptional cases are no more than 80 which is significantly fewer than the others. Thus, the records of those races are not up to date and the average finishing time is not accurate either.

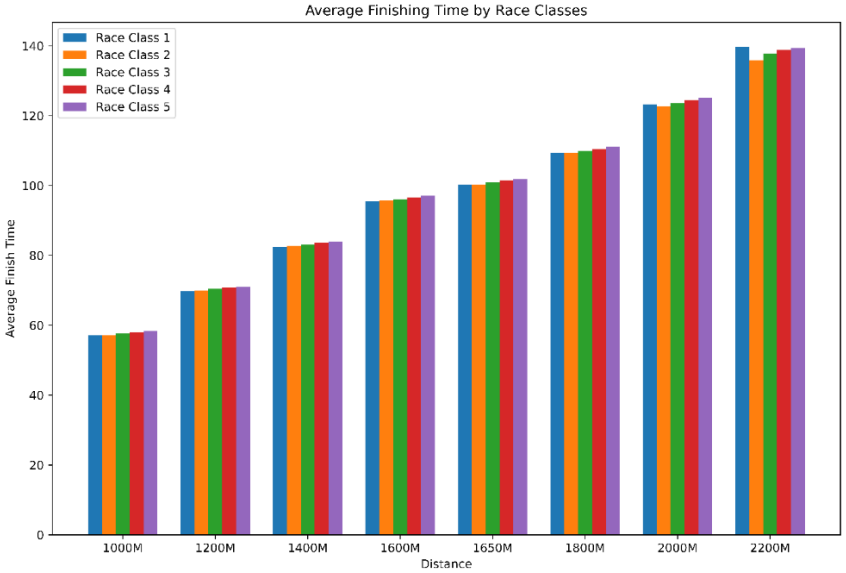

Figure 3.1 Relationship between Performance and Race Classes

An observation that the difference pf average finishing time between the race classes is as small as 0.5 seconds in the races with shorter distance and increases proportionally with the distance can be made. Therefore, race class can only affect the prediction of result moderately since the time difference is obvious overall.

3.3.2 Relationship between Performance and Rating

The rules of rating have been briefly introduced in section 1.3.2 and the previous section. In general, the rating of the horse would increase, remain unchanged or decrease after a race, depending on the place of the horse by the handicappers who is professional. If the horse has never participated in the horse racing in Hong Kong, rating of the horse will be assessed after participated in the race of Griffin race class or by the previous performance in the foreign races.

In each race, the horse number is assigned according to the descending order of the rating of the horses and the rating is equal to the actual weight (lbs.) that the horses need to carry during the handicap race. There would be horses with the same rating in the same race and the order of those horses would be decided by the automatic program and the maximum difference of rating in the same race is 20. The following graph shows the relationship between the rank of rating in a race and the final place of the horse.

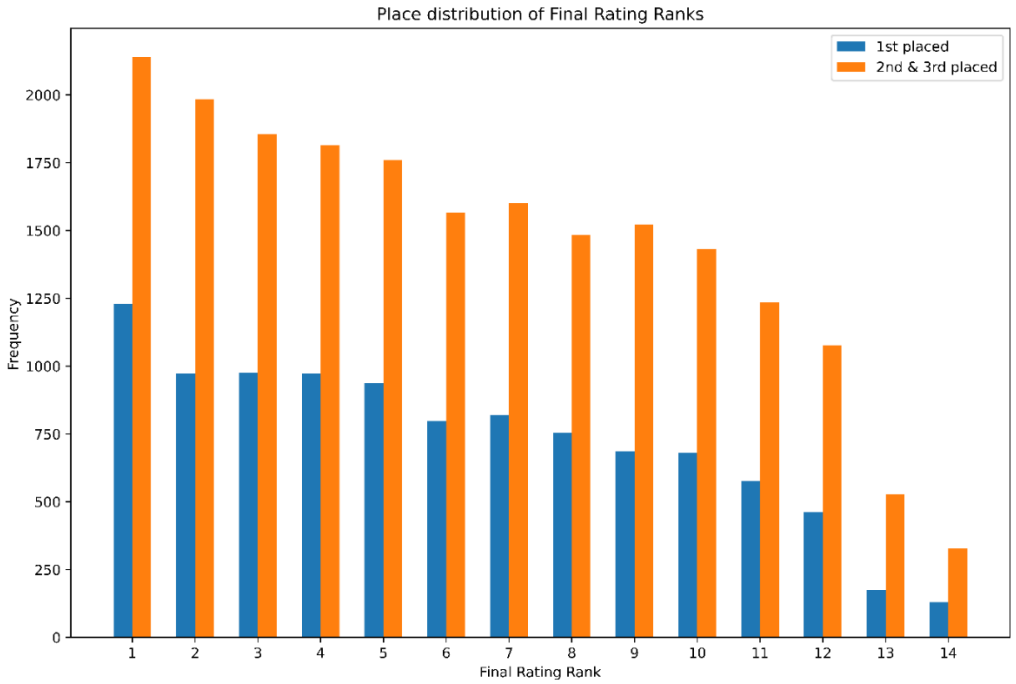

Figure 3.2 Relationship between Performance and Rating

The overall distribution of the place and the ranks of rating can be obviously observed that the horses with higher ratings in the race tend to perform better among the other horses in the same

race. It leads to the conclusion that the rating and the rank of rating does influence the finishing time of the horse.

In addition, for the horses with rank of rating 6 to 9, the trend fluctuates instead of decreasing continuously which may be caused by the same rated horses and thus, the performance is similar. Furthermore, there is a significant drop between the rating rank 12 and 13 which is the result of the rules that races with 14 horses can only be held in Sha Tin Racecourses, and thus, the number of races with 14 horses are fewer overall.

Although there are exceptions in the data, the overall distribution of the place is still able to illustrate the relationship between rating and place which is higher rating and carries heavier weights tends to perform better than the other horse in the same race

3.3.3 Relationship between Performance and Odds

The pari-mutuel local pool is adopted by the Hong Kong Jockey Club as mentioned in the section 1.3.3 which results in that to a certain extent the betting odds is able to reflect the public intelligence. The WIN betting odds would be computed by dividing the total pool after the Hong Kong Jockey Club taking 17.5% of it by total winning combinations. Thus, the betting odds of a horse would be lower if it is more popular or predicted to have a higher winning opportunity. The graph below shows the relationship between the final WIN odds ranking and the place.

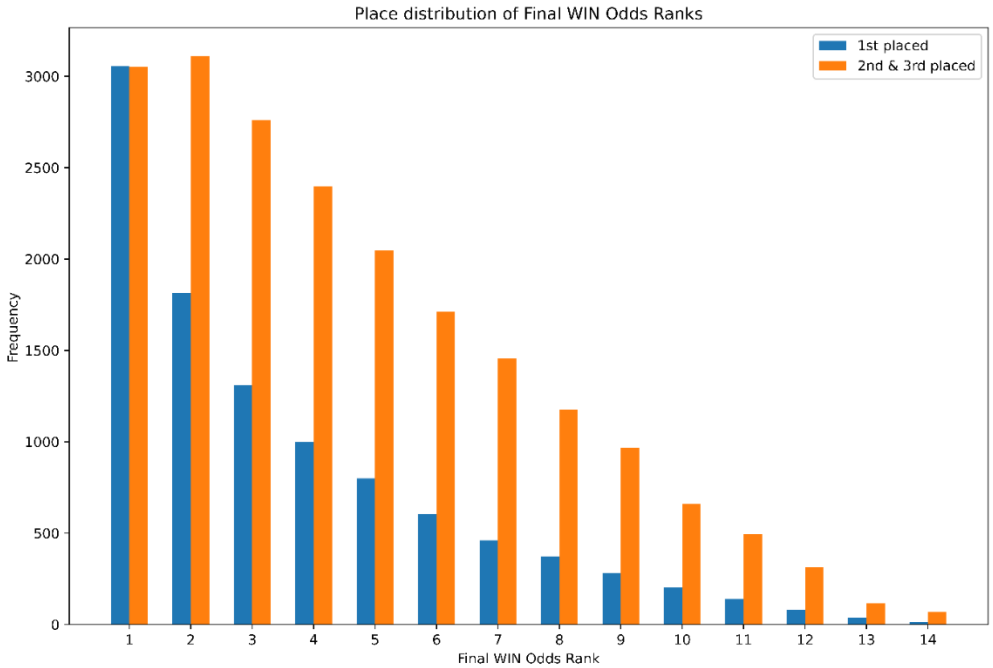

Figure 3.3 Relationship between Performance and WIN Odds

In the graph, the frequency of horses getting 2nd and 3rd place is added besides 1st place. A clear downtrend in the frequency of horses getting 1st place can be observed and the rate of change decreases along the final WIN odds ranking, while for the frequency of horses getting 2nd and 3rd place slightly increase at the beginning and then decreases with a relatively stable rate of change along the WIN odds ranking as well. Furthermore, the frequency of horses getting first 3 places obviously decreases along the WIN odds ranking.

From the graph, a conclusion that the betting odds can correctly reflects the winning probability of the horses, especially for the peak distribution of horses getting 2nd and 3rd place is at the second lowest WIN odds since gamblers would not wager WIN but only PLACE if the horses are not expected to get the 1st place but still have a higher expectation on their performance. Overall, as the total frequency of horses getting first 3 places decreases inversely proportional to the WIN odds ranking which shows that public intelligence could be an indicator for evaluating our prediction model.

3.3.4 Change of Average Finishing Time

There is a course standard times set by the Hong Kong Jockey Club which is the expected finishing time for horses in different race classes and different distances of race. The standard would be updated regularly which shows the performance of horses either improving or worsening in a particular race season. Thus, the following graph shows how the average finishing time changes over the years.

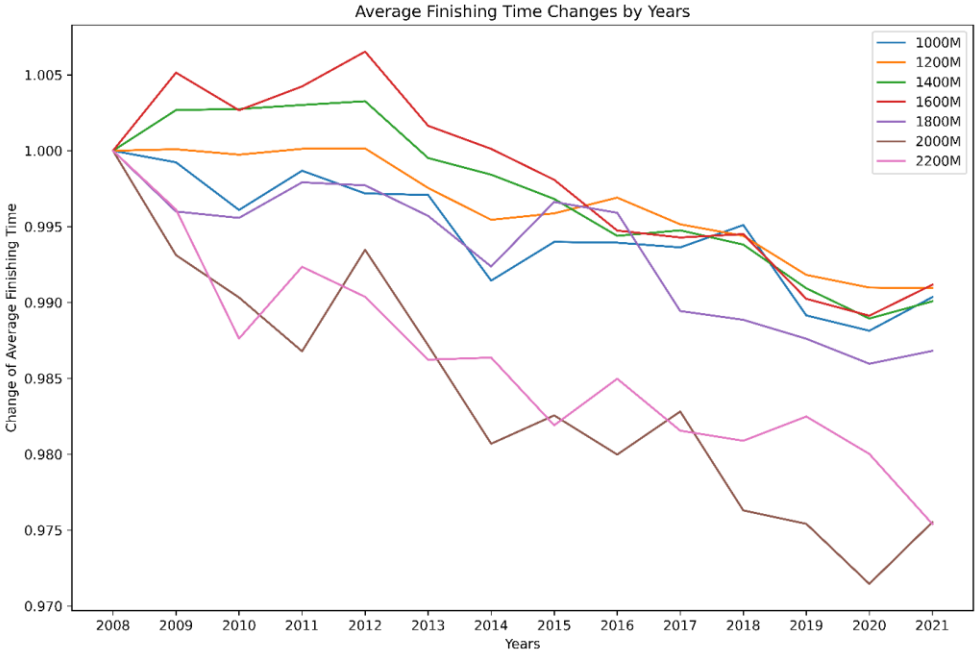

Figure 3.4 Trend of Overall Performance

The finishing time included both winning and losing horses are used for computing the average and the changes of the average finishing time are shown separately according to the distance of the races. The change of average finishing time is computed by dividing the data by the data in 2008, and thus, the change is with respect to 2008.

From the graph, a downtrend in the change of average finishing time starting from 2012 to 2021 is shown which clearly shows that the overall performance of the horse is improving. Especially for the race in 2000M and 2200M, the percentage of change exceeds 2.5%. For instance, the expected time for horses in race class 5 to complete 2200M is 135.8 seconds, and thus, there is an improvement of around 3 seconds over the years in this race.

Since the average finishing time of all horses is decreasing over years, and the training and testing data used in the time prediction model are split according to the race season, thus, it may impact the accuracy of the time prediction result.

3.4 Pre-processing of Data

3.4.1 Categorical Data

Since most of the algorithms do not support non numerical input data, as well as the library that we are using to build the random forest model, which is sci-kit learn, does not support the label data, and requires all input data to be numeric. Data encoding are needed to convert the string data such as the id of horse, race, and the gears etc. Furthermore, although some are features are in numeric type, for instance, the distance of the race, they are categorical data as well which requires encoding before building the model.

3.4.1.1 One-hot Encoding

The approach of one-hot encoding is to encode the categorical data with a new binary variable and remove the original data. In general, a categorical feature with k distinct values will be encoded as a feature vector of length k [52] [61]. While the number of features would be rapidly increased such that the efficiency of processing and training would be impacted, each encoded feature is represented by a distinct value which preserves the independence of the nominal variables. In addition, an all-zeros vector will be encoded if there is missing data in the categorical feature, and the mean of the target variable would be the output of the vector [52] [61]. The following tables show the data of the sex of the horse before and after encoding.

Table 3.8a Sex of Horses Feature Before One-hot Encoding

Index	Sex of horses
1	Colt
2	Filly
3	Gelding
4	Horse
5	Mare
6	Rig

Table 3.8b Sex of Horses Feature After One-hot Encoding

	sex_Colt	sex_Filly	sex_Gelding	sex_Horse	sex_Mare	sex_Rig
1	1	0	0	0	0	0
2	0	1	0	0	0	0
3	0	0	1	0	0	0
4	0	0	0	1	0	0
5	0	0	0	0	1	0
6	0	0	0	0	0	1

3.4.1.2 Ordinal Encoding

The approach of ordinal encoding is to encode the categorical data with numeric values such as 0, 1, 2. In general, the finite and distinct values of a categorical feature are mapped to a distinct numeric value [18]. The mapping of a feature value can be either customized or determined by the implementation of the encoder. The ordering relationship between values in a categorical feature can be maintained if a customized mapping is applied. The following tables show the data of race distance before and after encoding.

Table 3.9a Distance Before Ordinal Encoding

Distance
1000M
1200M
1400M
1600M
...
2000M
2200M
2400M

Table 3.9b Distance After Ordinal Encoding

Distance
0.0
1.0
2.0
3.0
...
6.0
7.0
8.0

3.4.1.3 Encoding of Gear

The gears equipped by the horses in a race are encoded by a customized approach, which is similar to the one-hot encoding. This approach is to first create columns for one gear only, while the number of the newly created columns does not match the number of distinct values in original gears columns, as each horse can equip more than one gear. A scheme is adopted for encoding instead of encoding with a new binary variable.

Table 3.10 Customized Encoding Scheme

Encoded variables	Description
0	Not equipped
1	Equipped for one or more consecutive races
2	Equipped for the first time since this race
3	Equipped the gear again since last unequipping the gear
4	Unequipped since this race

Instead of indicating the gear equipped by the horse only, the experience of equipping the same gear or not equipping any gear would be indicated. The reason to design this encoding scheme relates to the gears rule implemented by the Hong Kong Jockey Club that horses need to undergo a test or be proved they are suitable to participate in a race after equipping specific gears. This shows that equipping new gears for racing, horses would behave abnormally for gear changes which is a crucial factor affecting their performance. The following tables show the data of the gears equipped by horses before and after encoding.

Table 3.11a Gears of Horses Before Encoding

ID of the horse	Gears equipped
HK_2002_C230	-
HK_2002_C230	B
HK_2002_C230	B/TT
HK_2002_C230	B/TT
HK_2002_C230	-
HK_2002_C230	B
HK_2002_C230	PC

Table 3.11b Gears of Horses After Encoding

	PC	TT	SR	P	B	...	BO	V	CC
HK_2002_C230	0	0	0	0	0	...	0	0	0
HK_2002_C230	0	0	0	0	2	...	0	0	0
HK_2002_C230	0	2	0	0	1	...	0	0	0
HK_2002_C230	0	1	0	0	1	...	0	0	0
HK_2002_C230	0	4	0	0	4	...	0	0	0
HK_2002_C230	0	0	0	0	3	...	0	0	0
HK_2002_C230	2	0	0	0	4	...	0	0	0

3.4.2 Continuous Data

The continuous data differs from the categorical data since the former one is a type of numeric data which is possible to include infinite values. Although continuous data is not necessary to be encoded, preprocessing is still required to deal with some situations such as missing data which are not supported by the implementation of sci-kit learn. Additional data based on the continuous data are also created before building the model which provides extra information for the horse racing prediction.

3.4.2.1 Imputation of Distance Interval Data

The distance interval data is a group of features in a horse race record which includes the place, time, and horse length behind winner within a certain interval of the race. In contrast, the number of intervals varies depending on the total distance of the horse race which produces several missing data. Furthermore, inside the same distance interval column, existing values may represent different intervals which require preprocessing before building the time prediction model.

Table 3.12 Details of Distance Interval Data

	2400 – 2000M	2000 – 1600M	1600 – 1200M	1200 – 800M	800 –400M	Last 400M
1000M	/	/	/	(1000 – 800M)	✔	✔
1200M	/	/	/	✔	✔	✔
1400M	/	/	(1400 – 1200M)	✔		✔
1600M	/	/		✔	✔	✔
1650M	/	/	(1650 – 1200M)	✔	✔	✔
1800M	/	(1800 – 1600M)	✔	✔	✔	✔
2000M	/	✔	✔	✔	✔	✔
2200M	(2200 – 2000M)	✔	✔	✔	✔	✔
2400M	✔	✔	✔	✔	✔	✔

To deal with the missing data of the records which are a shorter distance, three different approaches are applied, and the evaluation of the approaches would be discussed in the latter chapters.

- **Constant Numeric Value**

The most intuitive approach to substitute the missing values is to replace them using a constant numeric value which does not exist as a value in other records. The constant numeric value used is –1 since either time, place or horse length cannot be a negative value, and thus, the substituted data is able to be computed and expected to be determined by the algorithm.

- **Attribute Mean**

Imputing data using the mean of the existing attribute is also known as "Most Common Attribute Value" substitution method (MC) [53]. In the previous work, it has been proved the performance of that imputing missing data imputed using MC is better than serval complex imputation algorithms such as Bayesian Principal Component Analysis (BPCA) and Support Vector Machines Imputation (SVMI) [40].

- **Data Calculated by Speed**

Since the distance interval data including time and place relates to the speed of the horse, missing data can be simulated based on the speed of the horse. This approach is to first compute the average

speed of the horse with the finishing time and the distance of the race, the time needed for a distance interval can then be computed. With the time of the distance interval, the place can be computed by grouping the race and ordering the time value.

3.4.2.2 Betting Odds Fluctuations

The data of the betting odds collected has shown the fluctuations of the betting odds, including both Win and Place, from the odds are released at noon of the day before the racing day to the final settled odds. As mentioned in section 3.2.4, the interval of the betting odds is different depending on the time left before the race starts and the fluctuation of the betting odds of a can still be reflected.

- **Efficiency of Horse Racing Betting Market**

The efficiency of the racetrack betting market has been proved in a previous study to be high which implies the betting odds would be an extraordinarily good estimate of the chance of winning, since the information and the betting odds are highly correlated, the message underlying the odds are needed to be retrieved, for instance, a significant change in the betting odds, which is possible to be a principal factor in betting strategy learning [55] [20].

Nevertheless, it is difficult and ambiguous to verify that to what extent the betting odds fluctuation can be determined as significant since the ranges of betting odds fluctuation are different for each horse.

- **Exponential Moving Average**

In order to design a standard for determining the significance of betting odds fluctuations of each horse, the concept of Exponential Moving Average (EMA) is adopted. In general, Moving Average (MA) can smoothen the fluctuation of data and be used to define the underlying trends [5]. MA is the average of previous data and EMA places more weight on the recent data and both are commonly used in the trading market as a signal of buying and selling.

The calculation of EMA is as follows:

$$EMA_t = (1 - SF) * EMA_{t-1} + SF * O_t \quad (9)$$

where SF $= \frac{2}{(n+1)}$, t denotes the timestamp and O_t denotes the odds of current timestamp t. The graph below shows a betting odds fluctuation and 2 different EMAs, which is 4-EMA and 8-EMA, computed from it. The difference between the EMAs is the weighting of the previous and recent data. The 4-EMA would be used throughout the project since the 4-EMA weighed more on the recent data, it is able to trace the closely which is suitable for the frequent and rapid changes in the betting odds which would produce fewer lagging signals of significant changes.

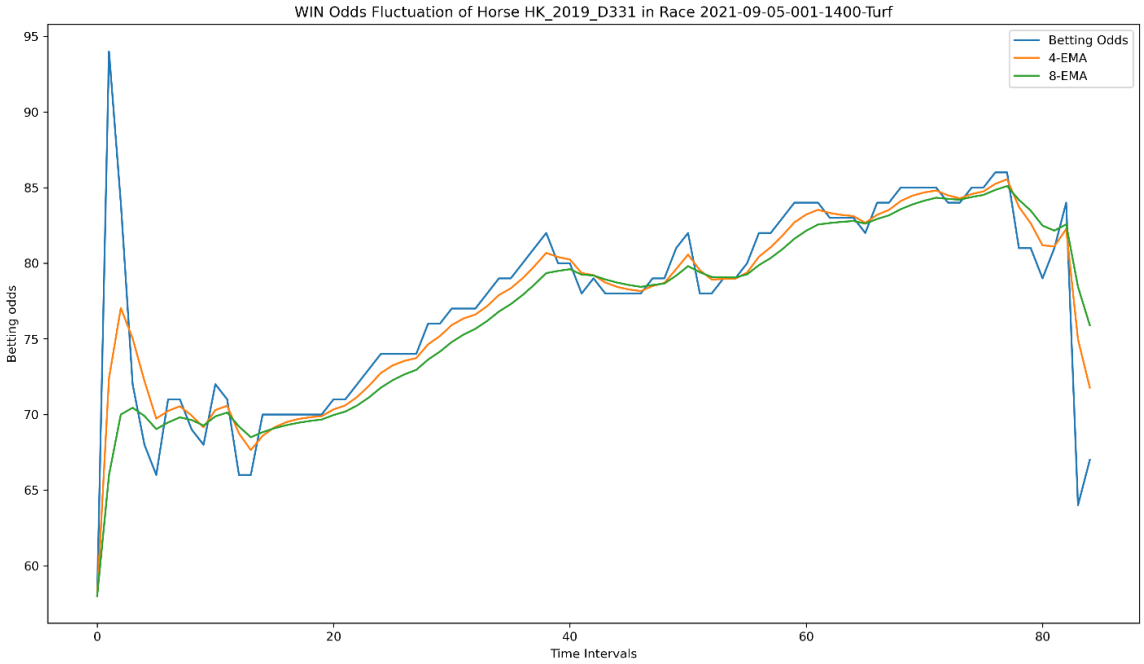

Figure 3.5 WIN Odds Fluctuation of a Random Horse

Since the pari-mutuel pool is adopted, the betting odds are dependent on the amount of bet which can only be finalized after the race starts. Gamblers tend to make wagers at the last moment such that the odds are closer to the final value which implies the fluctuation of odds during the last moment would be significantly influential. Thus, the EMA of only the last 10 records would be included in the datasets for training purposes.

3.4.2.3 Additional Features

Besides the data collected from the sources provided in section 3.1, additional features are computed from the data collected. The majority of the new features show and illustrate the previous performance of the horse and the difference of horse condition between the previous and current race since the past experiences of the horses are expected to be a key factor to determine the recent performance. All computed features are computed according to the order of race in order to avoid invalid data and shown in table 3.13

.

Table 3.13 Details of Additional Features

Features	Descriptions	Types	Sample
weight_diff	The difference of declared weight of the horse between last race and current race	Real Value	Positive or Negative Value
last_weight	The declared weight of the horse in last race	Real Value	Positive Value
last_rating	The rating of the horse in the last race	Real Value	Positive Value
last_place	The final place of the horse in last race	Categorical	1 – 14
count_{1-14}	The count on final place the horse got in the past	Real Value	Positive Value
last_pos_time_{1-6}	The time of the horse at different distance interval in last race	Real Value	Positive Value
last_pos_place_{1-6}	The place of the horse at different distance interval in last race	Categorical	1 – 14
last_speed	The average speed of the horse in the last race	Real Value	Positive Value
avg_rating	The average rating of the horse in all previous races	Real Value	Positive Value
avg_pos_time_{1-6}	The average time of the horse at different distance interval in all previous races	Real Value	Positive Value
avg_pos_place_{1-6}	The average place of the horse at different distance interval in all previous races	Real Value	Positive Value less than 14
avg_speed	The average speed of the horse in all previous races	Real Value	Positive Value
avg_finishtime	The average finishing time of the horse in all previous races	Real Value	Positive Value
win_odds_rank	The ranks of final win odds in the races	Categorical	1 – 14

3.4.3 Dropped columns

Training data can always be classified into relevant, irrelevant, and redundant, and if several features in the data collected are misleading or inappropriate for training the horse racing prediction model, the search space size would obviously increase especially in high dimensional data which creates more difficulties in processing data and obstacle the learning process [34]. Thus, those data are described in the following sections which are necessarily removed in order to increase the accuracy and performance of the model training process.

3.4.3.1 Redundant Features

The Uniform Resource Locator (URL) of the horse in the horse data is considered as redundant data since it is only indicating the official website of the horse which includes all details of the horse. Although the URL of each horse is distinct, the ID of the horse would be the intuitive choice for identifying each horse and using in the training process.

The age of the horse data is considered as redundant since there are missing records for retired horses. In addition, the age features only describe the current age of the horses instead of the exact age of the horses when they participated in the race. Although the age of the horse has been proved to be a considerable factor that affects the race earnings by investigating returns on betting by age group [13], the feature of age is necessary to be removed due to the fact of lacking data.

The length behind winner of the horse in the horse race data, which illustrates the distance between the horse and the first placed horse in unit of horse length, is considered as redundant. Length behind winner exists in both distance interval and full race features while the data is not able to reflect the exact distance as it is only an approximation. Granted that the accuracy of the length behind winner at different distance intervals is high enough for training, it is not necessary for predicting the finishing time of the horse since the exact distance is not required to be determined and the data may create noise for the training process.

3.4.3.2 Irrelevant Features

The name of the horse in the horse data is considered as irrelevant data since the horses would be given the same name. In contrast, having an identical name, horses are not expected to perform exactly the same. Thus, using the name of the horse as a training feature may mislead the model which would reduce the accuracy of the result.

The owner of the horse is irrelevant for the training process as owners of horses do not take a significant role in the process of horse training and the performance of horse mainly depends on the trainers and jockey. Thus, the owner of the horse has ignorable effort on the performance prediction of horse.

4. Horse Racing Prediction with Random Forest

4.1 Introduction

The earliest method of random forest is proposed by Tin Kam Ho in 1995 [27]. It is a tree-based machine learning algorithm that uses the idea of bagging (bootstrap aggregating) in ensemble learning and combines results from multiple trees instead of using a single tree. It still shares advantages of decision trees such as interpretability and no need for normalization while avoiding issues like easy overfitting and sensitivity to change of data which makes it a desirable choice for this project.

4.1.1 Classification and Regression Tree

The Classification and Regression Tree (CART) is one of the algorithms in decision tree learning. A classification tree outputs a categorical value while a regression tree outputs a numeric value. The process of constructing a classification tree or a regression tree is similar which is based on the first published classification tree algorithm THAID [39] [1] [50]. The THAID searches exhaustively over all variables (training data) X and splits X into 2 subsets with the minimum total impurity as its child nodes. The process of splitting is recursively applied on each child node until a stopping criterion is satisfied.

For the classification tree, Gini index which is a generalization of binomial variance is used as the impurity function and is given by the following equation [26]:

$$Gini(t) = 1 - \sum_{i=1}^{j} P(i|t)^2 \tag{10}$$

where j is the number of classes, t is the subset of instances for the node and $P(i|t)$ is the probability of selecting an element of class i from the node's subset.

For a regression tree, the impurity of each node would be compared by the sum of squared residuals and is given by the following equation:

$$SSR(t) = \sum_{i=1}^{n} (N_i - \overline{N_t})^2 \quad (11)$$

where N_i is the targeted value of input X, t is the subset of instances for the node and $\overline{N_t}$ denotes the mean of all elements in subset t .

4.1.2 Bagging

Bagging is one of the earliest and simplest trees ensemble algorithms can be applied which refers to bootstrap aggregating which means first bootstrapping the datasets and use the aggregate to make a decision [11].

Consider a training dataset $L = \{(y_n, x_n)\}$ where y is targeted value or class label of x and $n = 1, \dots, N$. The process of bootstrapping datasets refers to create a new set of training datasets $\{L^{(B)}\}$ by replicating N data from L randomly and with replacement for each dataset $L^{(B)}$, and thus, each data (y_n, x_n) may not appear or be repeated in a particular dataset $L^{(B)}$.

The process of aggregating varies depending on the type of y instead of returning only a prediction from a single tree. Assume that the predictor function $p(x, L^{(B)})$ which accepts input x and predict y by each of training dataset $L^{(B)}$. If the type of y is numeric, the output would be the average of $\{p(x, L^{(B)})\}$. Whereas if y is a categorical value, the output would be decided by voting where the output y appeared most would be selected.

Combining bootstrapping and aggregating, bagging is beneficial as the variance is reduced and has been proved that the mean square error would not be increased but often be reduced as well [24].

4.1.3 Ensemble and Random Sampling

The Random Forest is constructed by numerous of distinct CART and uses the idea of tree bagging which create different CARTs with bootstrapped training datasets and aggregating the prediction by averaging or voting [12]. Furthermore, random input selection is adopted in constructing random forest. During the process of CARTs construction, instead of searching every variable X exhaustively and finding a best split using the impurity function, only a small random subset of variables with a size of $\log_2(n) + 1$ which greatly increases the efficiency of the node splitting process [24].

4.2 Modelling Approach: Random Forest

The approach to make a prediction on horse racing is to apply random forest for training the preprocessed data. The data would be separated into training and testing datasets. The training dataset would be fed into the random forest, and it would be randomly sampled as bootstrapped data. The corresponding regression tree would be trained and the average of prediction from all regression trees would be aggregated as the output. After the model has been built, testing dataset would be fed into the regressor and predict on the finishing time. Then the result would be used in evaluation of the model as well as analysis and also for the multi-armed bandit training.

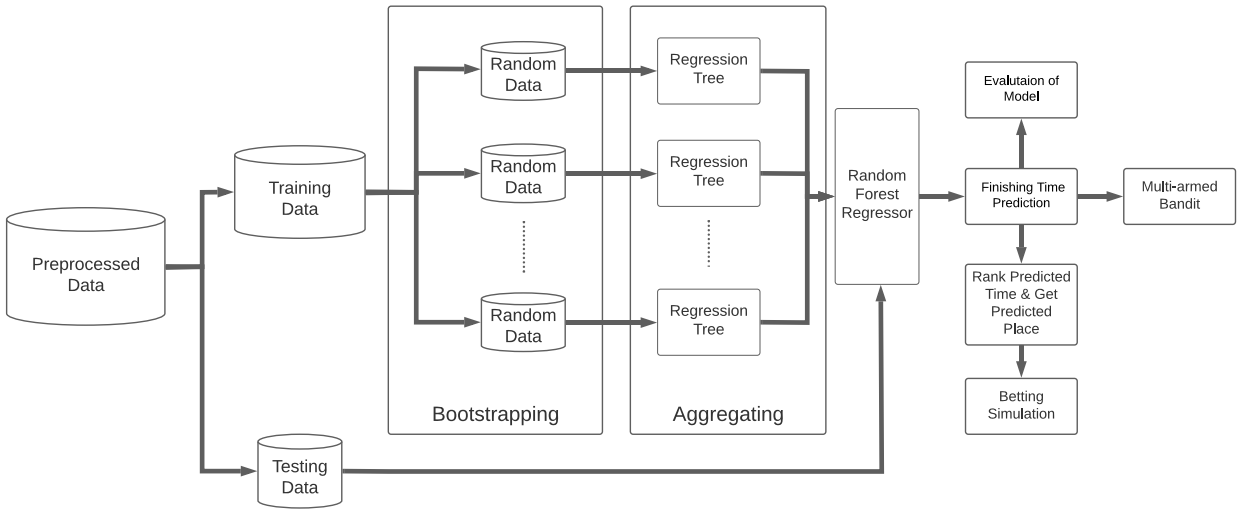

Figure 4.1 Data Flow Chart of Time Prediction Procedure

4.3 Motivation

Considering the CART, from the 2 impurity functions mentioned in section 4.1.2, a convergence of the predict value, no matter of classification or regression, can be obtained if the depth of the tree is high enough. This implies that the accuracy of prediction of the training datasets would be high which may be suitable for our purpose to predict an accurate finishing time of each horse.

In contrast, a single decision tree tends to overfit the fixed training datasets if the depth of the tree is high which results in low accuracy in testing datasets. Tree pruning may increase the accuracy of testing data while consuming the accuracy of training data [27].

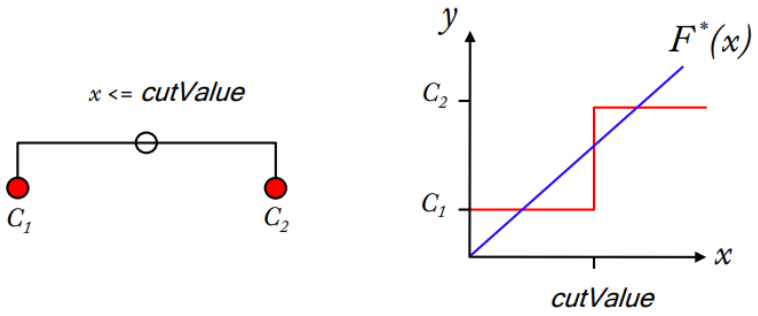

Figure 4.2 A 2-terminal Node Approximation to a Linear Function [24]

Another limitation of the single decision tree is that it is a discontinuous piecewise constant model. For every split, information would lose since average values are predicted as shown in figure 4.2. Thus, a single decision tree requires number of splits to accurately predict a trend.

In order to make a more accurate prediction of horse racing results, instead of using a single CART, random forest would be a better option since random forest adopts the idea of tree bagging. Random forest has been proved that it does not overfit as the number of trees are increasing as well as the convergence of its predictions by Strong Law of Large Numbers [12]. The preciseness of the experiment has to be controlled carefully since the difference of finishing time of the first few horses in a race is as small as 0.5 seconds. In addition, random splitting gives a huge benefit to the process of experiment as the time on tree constructing would be reduced. Since the training and testing data consists of thousands of features after encoding, random forest is a suitable option for our project.

4.4 Result and Analysis

4.4.1 Time Prediction Procedure

The purpose of the model training is to predict the finishing time of each horse in a race and thus, the intermediate output can be used to figure out the features that would have significant influents on the prediction of the result. Furthermore, a predicted place would be assigned to each horse by ranking the time prediction and figure out the difference between our prediction and actual records.

4.4.1.1 Input Data

The training data and testing data would be split according to the race season in order to have a clearer illustration on result analysis. The training data contains 108107 records from season 2008/2009 to 2018/2019 while the testing data contains 18002 records of seasons 2019/2020 and 2020/2021.

Table 4.1 Features of Input Data

Features	**Types**	**Encoding Methods**
raceclass	Categorical	Ordinal
tracktype	Categorical	One-hot
racktrack	Categorical	One-hot
course	Categorical	One-hot
country	Categorical	One-hot
importtype	Categorical	One-hot
sex	Categorical	One-hot
colour	Categorical	One-hot
going	Categorical	One-hot
jockey	Categorical	Ordinal
trainer	Categorical	Ordinal
horseid	Categorical	Ordinal
dam	Categorical	Ordinal
sire	Categorical	Ordinal
damsire	Categorical	Ordinal
distance	Categorical	Ordinal
draw	Categorical	Ordinal
rating	Real Value	/
rating_rank	Real Value	/
last_rating	Real Value	/
avg_rating	Real Value	/
last_place	Real Value	/
winodds	Real Value	/
win_odds_rank	Real Value	/
actualweight	Real Value	/
declaredweight	Real Value	/
gear	Categorical	Customized Encoding
raceidseason	Real Value	/
count_{1 – 3}	Real Value	/
weight_diff	Real Value	/
avg_finishtime	Real Value	/
avg_pos{1 – 6}_pos	Real Value	/
avg_pos{1 – 6}_time	Real Value	/
last_pos{1 – 6}_pos	Real Value	/
last_pos{1 – 6}_time	Real Value	/

For the features in the table 4.1, such as jockey, trainer, horseid, dam, sire and damsire which has no ordering relationship between each value, one-hot encoding is supposed to be used as mentioned in section 3.4.1 while ordinal encoding is used. The reason for this is that the total number of distinct combinations for these features exceeds 6600. If one-hot encoding is applied, more than 6700 features would be input for training model which would greatly affect the procedure of random forest model training. As for every feature, there are only 2 possible values – 0 and 1, it is possible that the structure of a decision tree is identical to the following picture that tends to grow on the same side and performs poorly.

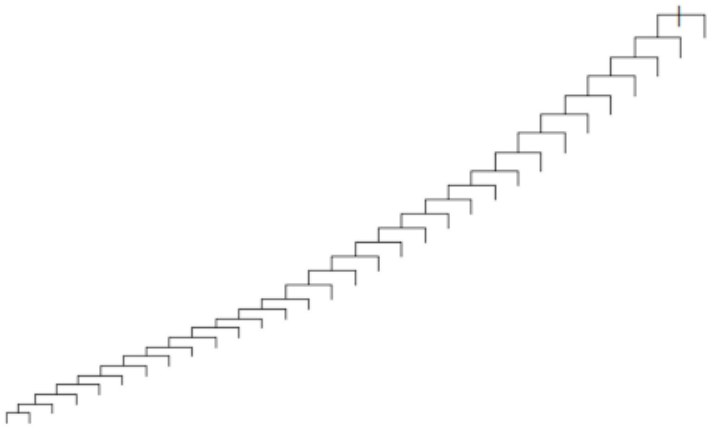

Figure 4.3 Possible Structure of Decision Tree [35]

As mentioned in section 3.4.2.1, there are several approaches used to imputing the distance interval data which are using a constant value, mean or the speed of the horse. Thus, all three approaches are separately tested in order to figure out which one gives the most accurate and suitable outcome and selected for the betting strategy training.

4.4.1.2 Training

The parameters for the random forest training are optimized by cross-validated grid-search over a parameter grid exhaustively which would be shown below.

- The number of decision trees is 256
- The maximum depth of each decision tree is 13.
- The minimum number of samples required to split an internal node is 2.
- The metrics for comparing the quality of each node split is mean squared error

After the random forest training, the predicted finishing time of each horse would be compared to other horses which participated in the same race, and thus, the predicted place would be ordered by the ascending order of the predicted finishing time. The following is a sample result for a

randomly selected race. The place feature refers to the actual place of the horse and pred refers to the predicted finishing time by the model.

	horseid	raceid	distance	winodds	place	pred
17995	5,265.00	2019-09-01-001-1600-Turf	3.00	2.20	1	95.84
17993	4,296.00	2019-09-01-001-1600-Turf	3.00	7.00	5	95.85
17994	4,268.00	2019-09-01-001-1600-Turf	3.00	5.70	4	95.86
17996	5,186.00	2019-09-01-001-1600-Turf	3.00	4.90	2	95.89
17999	4,302.00	2019-09-01-001-1600-Turf	3.00	18.00	3	96.16
18000	4,982.00	2019-09-01-001-1600-Turf	3.00	21.00	9	96.17
18001	4,809.00	2019-09-01-001-1600-Turf	3.00	19.00	6	96.21
17998	4,845.00	2019-09-01-001-1600-Turf	3.00	14.00	8	96.33
17997	5,103.00	2019-09-01-001-1600-Turf	3.00	50.00	7	96.35

Figure 4.4 Sample of Output Data

4.4.2 Structure of the Tree and Tree Path

The following record is one of the testing data which is used to illustrate the decision tree path and the corresponding decision tree. Since the number of features of the record are too large and the structure of the decision is too wide, thus, only partial of the record and decision tree would be shown.

	avg_finishtime	avg_pos6_place	avg_pos6_time	avg_rating	declaredweight	distance	draw	going_GY	horseid	raceidseason	rating	win_odds_rank
17994	96	4.2	23	36	1.1e+03	3	3	0	4.3e+03	1	37	3

Figure 4.5 Sample Data for Demonstration of Decision Path

```
decision id node 0 : (X_test[1, 'distance'] (= 3.0) > 2.5)
decision id node 3982 : (X_test[1, 'distance'] (= 3.0) <= 5.5)
decision id node 3983 : (X_test[1, 'distance'] (= 3.0) <= 4.5)
decision id node 3984 : (X_test[1, 'distance'] (= 3.0) <= 3.5)
decision id node 3985 : (X_test[1, 'rating'] (= 37.0) <= 63.5)
decision id node 3986 : (X_test[1, 'win_odds_rank'] (= 3) <= 9.5)
decision id node 3987 : (X_test[1, 'horseid'] (= 4268.0) > 2408.0)
decision id node 4075 : (X_test[1, 'going_GY'] (= 0.0) <= 0.5)
decision id node 4076 : (X_test[1, 'raceidseason'] (= 1) <= 688.5)
decision id node 4077 : (X_test[1, 'avg_rating'] (= 36.0) <= 44.038461685180664)
decision id node 4078 : (X_test[1, 'rating'] (= 37.0) <= 49.5)
decision id node 4079 : (X_test[1, 'horseid'] (= 4268.0) > 3718.0)
decision id node 4083 : (X_test[1, 'avg_pos6_place'] (= 4.166666666666667) <= 9.125)
```

Figure 4.6 Decision Path of the Sample Data

The root and first few nodes split the data by the distance value which is intuitive, and the tree branch after the 4th nodes would include the races that only have the same distance, which shows that the decision tree is able to figure out the finishing time of races with same distances are close.

In addition, rating related data and the WIN odds ranking have been split for three times and once respectively which illustrates that the decision tree can find out the relation between rating or WIN odds ranking and the finishing time or place as mentioned in section 3.3. The id of the horse has also been split for twice which implies that the model understands that different horses have difference performance and correspondingly predict the finishing time.

Furthermore, both the index of race in a race season and the going of the rack track has been split for once which demonstrates that the tree makes decision based on the environmental factors as well.

The following diagram shows parts of the decision tree structure which includes the last 4 nodes in the tree path above.

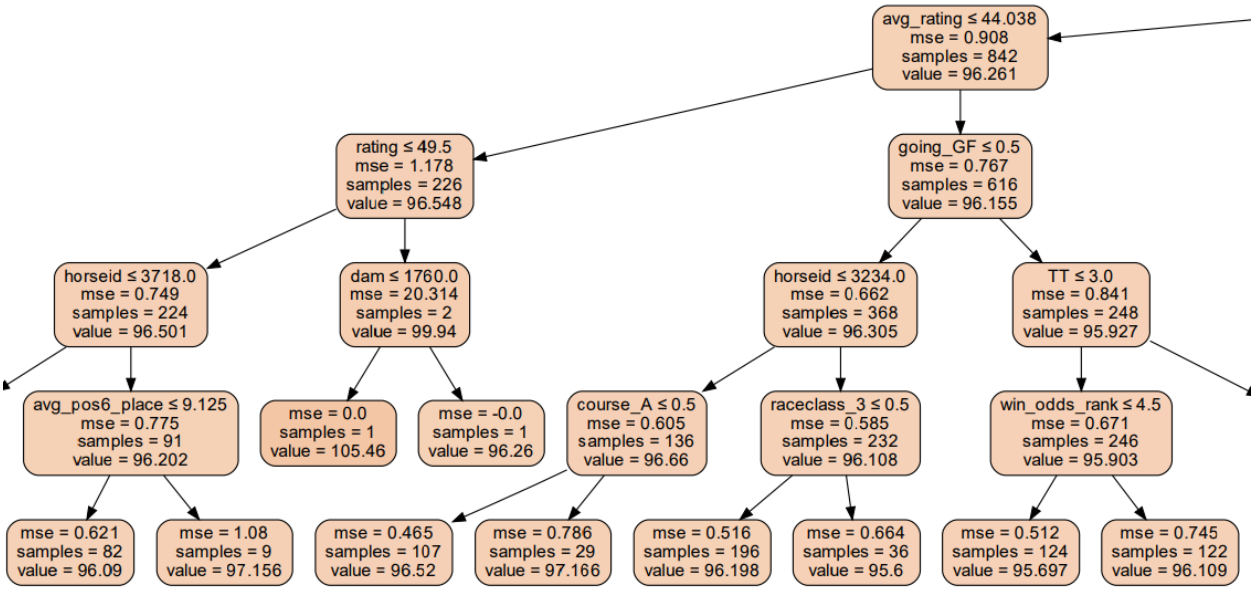

Figure 4.7 Partial Structure of Decision Tree

From the structure of the diagram, the predicted finishing time is 96.09 seconds, and the actual finishing time of this record is 95.68 seconds, which has a difference in 0.41 seconds. In general, the leaf nodes do not contain only a few samples which shows the training process is successful and it is not overfitting the training dataset.

4.4.3 Importance of Features

- **Impurity-based Feature Importance**

The impurity-based importance is used to measure to what extent the features affected the nodes and structure of decision tree. As mentioned in section 4.1, each internal node is split according to the impurity of the split datasets, and the feature which has the lowest impurity after splitting would be chosen as the feature for node splitting.

The impurity-based importance of each feature is computed by first measuring the mean and standard deviation of accumulation of the impurity is reduced by splitting on this feature within each decision tree, then computing the average of impurity decreased over all decision trees in the random forest [64]. Thus, the higher the impurity-based importance of a feature is, the greater number of nodes use the feature for splitting.

In contrast, the impurity-based feature importance can be misleading in extreme case if there exist features which have a high cardinality or many unique values such as id, since the sample of each record is not adequate. Thus, the permutation importance, which will be introduced in the next section, is needed for verifying those features.

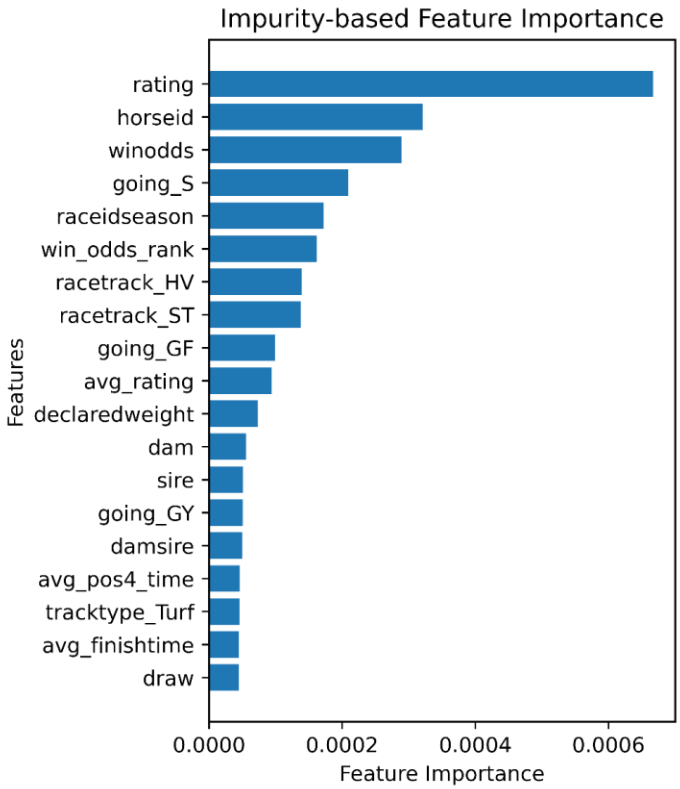

Figure 4.8 Features with Top 20 Impurity-based Feature Importance

Figure 4.8 shows top 20 features which is sorted according to their impurity-based importance in descending order. In the graph, the importance of distance has been hidden as the importance value reached 0.99631 which is significantly greater than all other features. If the important of distance is not hidden, all other bars would be compressed which is not meaningful to display the chart. It is intuitive that distance has the extremely high importance since the finishing time is clearly depending on the distance of the race.

Besides, rating and odds related features occupied 4 out of 20 which matches the analysis in section 3.3 regarding the existing relationship between rating or odds and final place. For declared weight and draw of the horse, in LYU1603 they have been proved to have a correlation with win percentage which can explain the importance of them are included in top 20 [15].

Interestingly, there are 6 out of 20 features related to the racetrack, type of track and going which are all environmental factors and the raceidseason, which refers to the index of race in each season and can be treated as a seasonal indicator, has the 6^{th} highest importance. Thus, the racing environment could influent the performance of the time prediction significantly.

Lastly, although id, dam, sire and damsire of the horse are features which have a high cardinality, they could not be verified using the impurity-based feature importance as bias maybe existed.

- **Permutation-based Feature Importance**

The permutation-based feature importance is used to measure to what extent the feature affects the performance of the model if the feature is removed from the data. The procedure of computing permutation-based importance is first, define a baseline metric, which is R^2 score, to evaluate the model on given dataset. Then, permute each feature from the dataset and evaluate the mode with the same metric again and figure out the difference between the baseline metric and the new metric computed after each feature column is permuted.

The permutation-based importance can verify the high cardinality features that maybe mislead by the impurity-based importance. Since the former option evaluates the whole model again after removing a complete feature which is estimating the negative impact brought by removing a feature, instead of just investigating to what extent the feature can enhance the performance of the model. However, the permutation-based feature importance has been proved that it would be misleading if there exist features which are significantly correlated which would result in returning low importance for possible features are actually important [54] [6] [49].

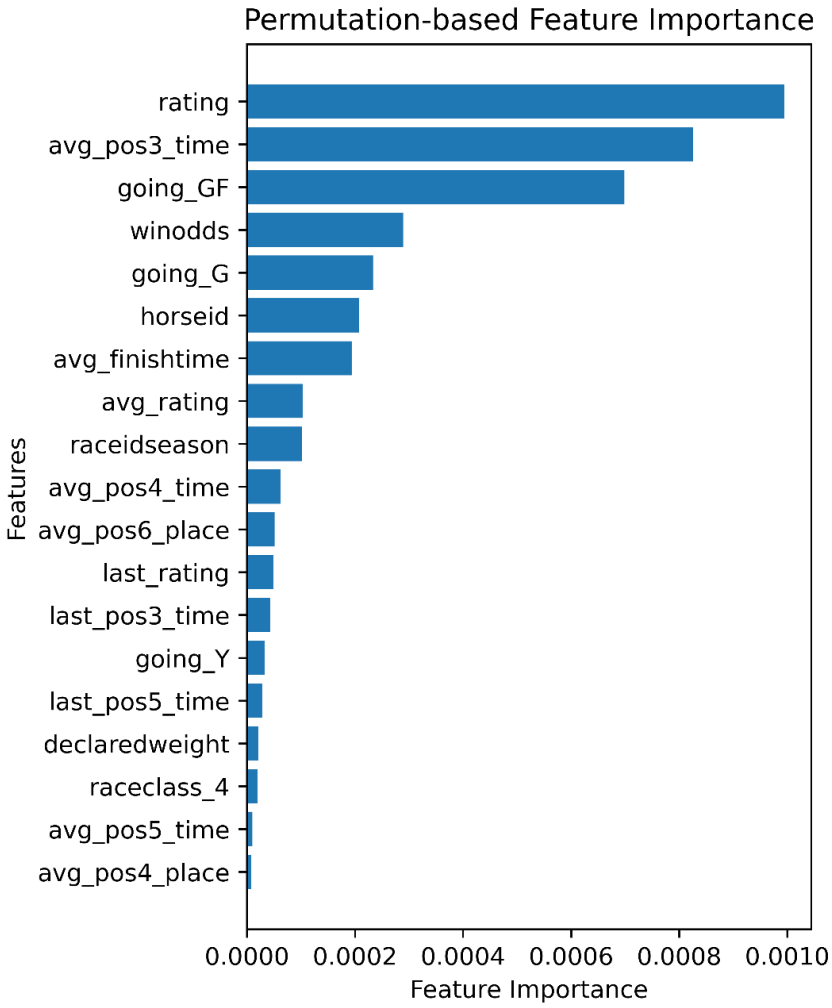

Figure 4.9 Features with Top 20 Permutation-based Feature Importance

Similarly, figure 4.9 displays the top 20 permutation-based feature importance, and the importance of distance is removed since it has reached the value of 2.0142 which is also significantly greater than importance of other features. Besides, the ranking of rating, going related and WIN odds are correspondingly high which shows that they are important features in the training process. For those features which have high cardinality, only the id of horses is included in both of the list of top 20 impurity-based and permutation-based importance which verifies that it has positive effect to the time prediction.

Furthermore, the major difference in the permutation-based feature importance is that more distance interval data, including both performance of average and previous race, are in the list of top 20 which shows that to a certain extent the place and speed of a particular interval would affect the time prediction model. Thus, it implies that the model the has been trained to figure the racing style of the horse would affect the performance of the horse.

4.4.4 Partial Dependence of Features

The partial dependence shows the marginal effects that a set of input features have on the target response of the machine learning model which is defined as the following [23][25][60].

$$pd_{X_S}(x_S) \approx \frac{1}{n_{samples}} \sum_{i=1}^{n} f\left(x_S, x_C^{(i)}\right) \tag{12}$$

where $x_C^{(i)}$ are the values of the i^{th} sample in the set of input features that are not interested and x_s is the value of feature of the input feature to be used to compute partial dependence, f is the predict function and $n_{samples}$ is the total number of data.

The partial dependence plot describes the changes of the prediction result against the value of the feature which can show whether the relationships between the feature chosen, and the prediction are linear, monotonic, or even more complex [23][25][60]. For numerical features, the plot of a selected feature is constructed by finding the partial dependence with all possible values within the range of that feature while for categorical features, the values used are all possible categories of that feature.

The partial dependence plot is intuitive, and the interpretation is clear since it can virtualize how each feature influences the result correspondingly while the graph would be misleading if the distribution of the feature is omitted since there would be an over-interpretation of the region that has almost no data. [17] Thus, rugs would be plotted on the graph indicating the distribution of the feature. In this section, the partial dependences of 5 features are selected to be observed based on the rank of their feature importance.

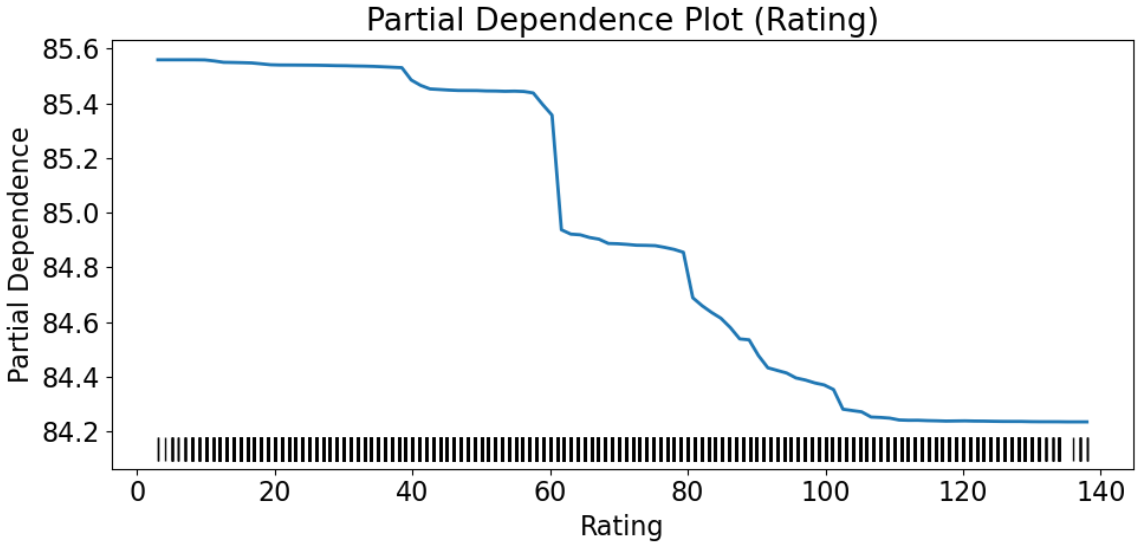

Figure 4.10 Partial Dependence Plot of Rating

Figure 4.10 displays the partial dependence plot of the rating feature which has the second highest feature importance mentioned in section 4.4.3. It clearly shows that the rating of horses is inversely proportional to the finishing time of the horse and the distribution of the rating is even in the data, and this matches the data analysis of the relationship between race class and finishing time in section 3.3.1. Besides, there are significant intervals indicating the boundaries of different race classes stated in section 1.3.1, which shows that our random forest model predicts the finishing time of horses by using the rating feature correctly.

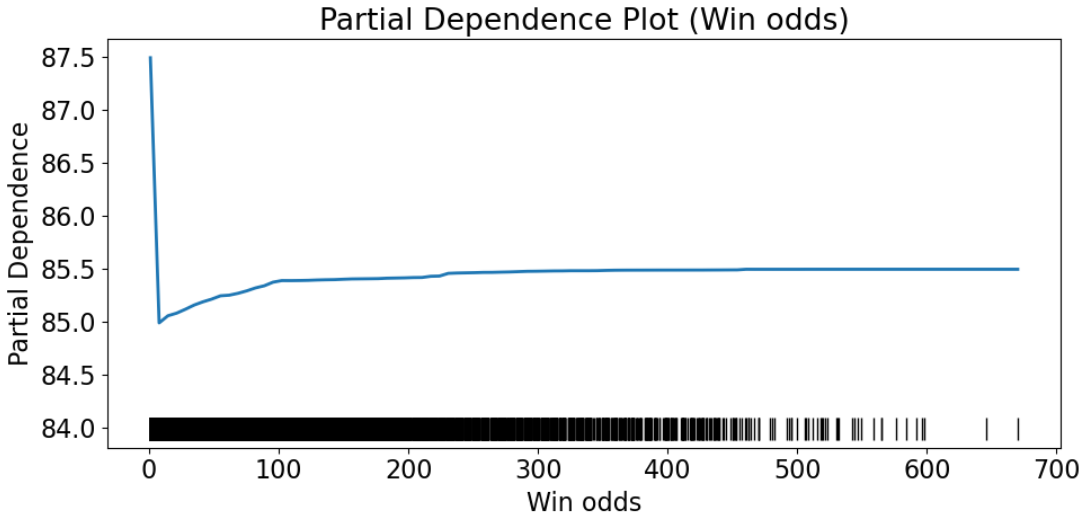

Figure 4.11 Partial Dependence Plot of Win odds

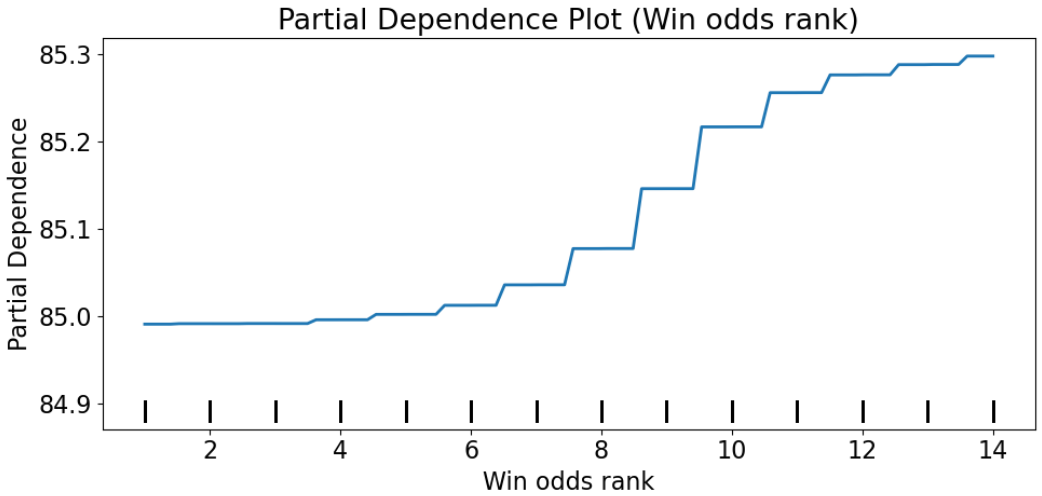

Figure 4.12 Partial Dependence Plot of Win odds rank

Figure 4.11 shows the partial dependence plot of the win odds feature which has one of the highest feature importance. From the graph, the partial dependence rapidly drops within the range of 0 and 10 while it increases after the significant drop and tends to be flat afterward. The reason is that the win odds of horses in different races can be extremely low while the races are not in the identical setting, for instance, the race class, track type and going are different which may affect the performance of the horses. Besides, the partial dependence tends to be flat when the win odd is greater than 200 showing that the larger values of win odd have no considerable influence on the time prediction since the distribution beyond the win odds value of 400 is sparser. Thus, the partial dependence is not able to reflect the real situation that the win odd of a horse increases with the finishing time for most cases.

Figure 4.12 shows the partial dependence plot of the rank of win odds in each race in ascending order. From the graph, an increasing curve is plotted, and partial dependence significantly increases when the win odds rank is 6. This represents that the rank of win odds is proportional to the predicted finishing time which is different from figure 4.11. Thus, it is obvious that our random forest model can give a prediction based on the win odd ranking of each horse.

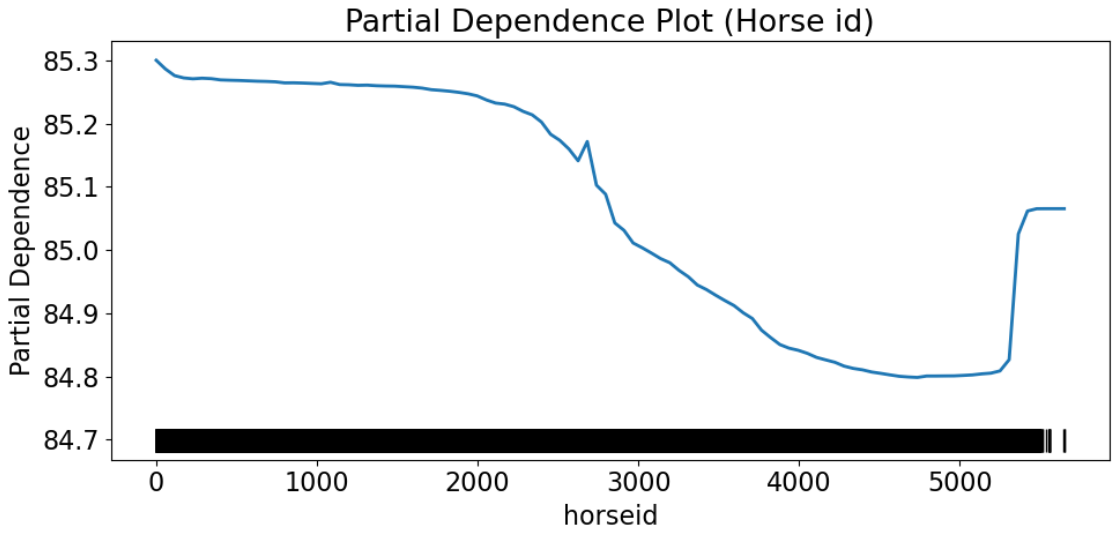

Figure 4.13 Partial Dependence Plot of Horse ID

Figure 4.13 displays the partial dependence plot of the horse id feature and horses with larger horse id represents that they are registered and participated in races more recently than other horses. From the graph, an obvious decreasing trend of partial dependence has been shown for the majority of horse id values which describes that the average finishing time of horses decreases along the time and matches the data analysis in section 3.3.4. While there is an increasing trend starting from horse id 5200 and one of the reasons may be the distribution of the horses within the range are

more discrete than before which leads to a lack of data to get a more general result for that range of horse id.

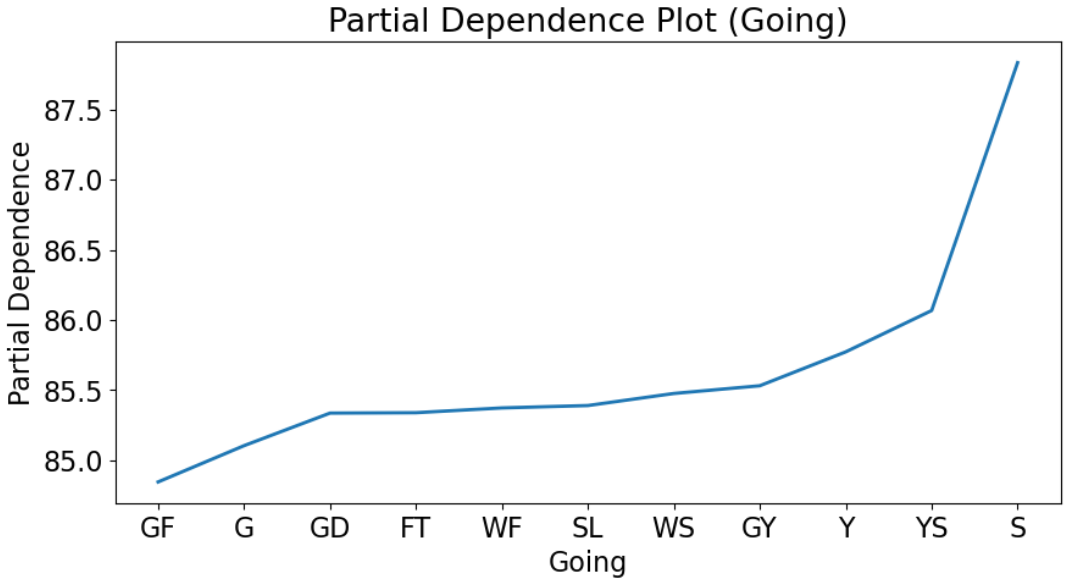

Figure 4.14 Partial Dependence Plot of Going

Figure 4.14 shows the partial dependence plot of the going of the course from all races including both Turf and All Weather Track. From the graph, an increasing trend overall has been displayed and the order of going matches the order described in section 3.3.1. Among all types of going, GD, FT, WF, SL, and WS are the going types of All Weather Track and they gave a similar partial dependence while other going types belong to Turf Track and there is a significant increasing trend shown for these types. Thus, going types belonging to All Weather Track have less influents on the time prediction result than going types belonging to Turf Track.

4.4.5 Evaluation Metrics

The mean squared error of the predicted and the actual finishing time, which is defined as the following, is computed to evaluate the accuracy of the overall time prediction [63].

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2 \quad (13)$$

where y_i is the actual finishing time in the testing data and $\hat{y}_i$ is the predicted value from the random forest model of the i^{th} data, $n_{samples}$ is the total number of testing data. When the value of mean square error is closer to 0, it represents the overall differences between the actual and the predicted finishing time are smaller and vice versa.

The explained variance score is defined as the following equation which is used to measure the discrepancy between the model and data [63].

$$explained\ variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}} \quad (14)$$

where y is the actual finishing time and $\hat{y}$ is the predicted finishing time from the mode, Var is variance. When the explained variance score is closer to 1, it represents the model and data are strongly associated with each other which has a betting prediction, while if the explained variance score is lower or is even a negative number, the association is weak.

- The mean squared error of the predictions is 2.2649 seconds.
- The explained variance score of the model is 0.99388.

In terms of model training, the time prediction model is successfully trained since the explained variance score shows that the model has a strong association with the actual data and the data can be well fitted into random forest model.

In terms of accuracy of finishing time prediction, the common time differences between the first and second are around 0.1 to 0.3 seconds and the mean squared error is significantly greater than that. In contrast, by considering the decreasing trend in the average finishing time stated in section 3.3.4, and the limitation of the random forest that it is not able to accurately predict the unseen data, which is not in the range of training data, the accuracy of finishing time prediction is still acceptable.

4.4.6 Difference between Actual and Predicted Place

Since the random forest only predicts the finishing time of each horse which is not meaningful to the betting strategies training, the predicted place of the horses needs to be ranked by the order of predicted finishing time in the same race.

	horseid	raceid	place	winodds	pred	pred_place	place_difference
17995	5265.0	2019-09-01-001-1600-Turf	1	2.2	95.84	1	0
17996	5186.0	2019-09-01-001-1600-Turf	2	4.9	95.89	4	2
17999	4302.0	2019-09-01-001-1600-Turf	3	18.0	96.16	5	2
17994	4268.0	2019-09-01-001-1600-Turf	4	5.7	95.86	3	-1
17993	4296.0	2019-09-01-001-1600-Turf	5	7.0	95.85	2	-3
18001	4809.0	2019-09-01-001-1600-Turf	6	19.0	96.21	7	1
17997	5103.0	2019-09-01-001-1600-Turf	7	50.0	96.35	9	2
17998	4845.0	2019-09-01-001-1600-Turf	8	14.0	96.33	8	0
18000	4982.0	2019-09-01-001-1600-Turf	9	21.0	96.17	6	-3

Figure 4.15 Sample Race with Predicted Places

The accuracy of place prediction can be measured by the distribution of the difference between the actual and the predicted place computed by the finishing time prediction which is shown in the following figure.

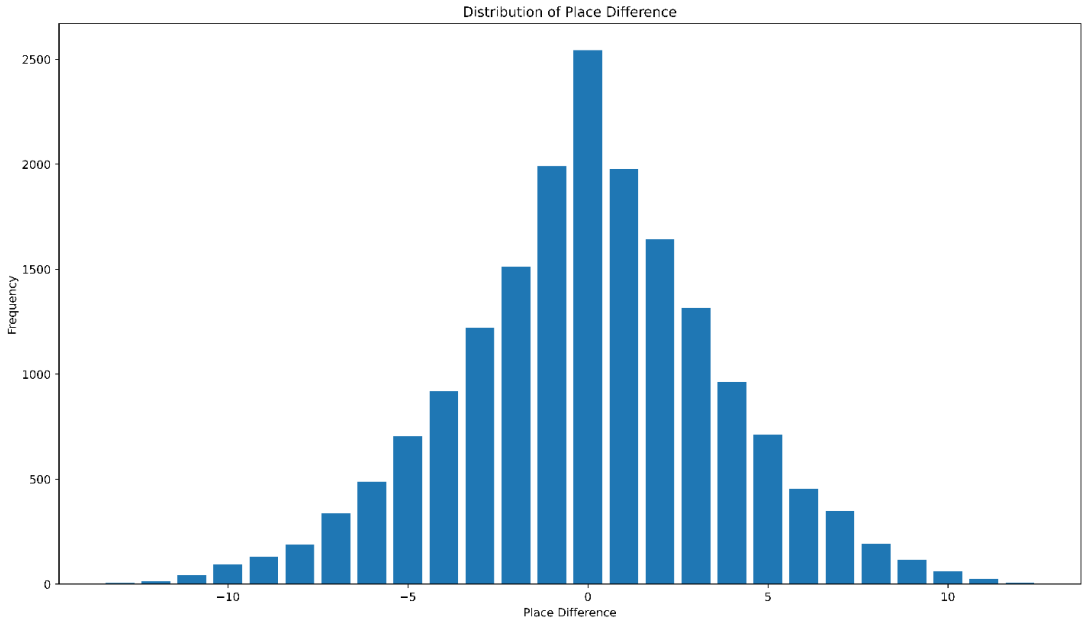

Figure 4.16 Distribution of Place Difference

The place difference is calculated by subtracting the actual place from the predicted place, and thus, both positive and negative numbers would be the result. The negative numbers mean the predicted place of the horse is overestimated and vice versa, and 0 means the prediction is correct.

From the graph, the value of place difference is concentrated at 0 and the distribution decreases in both positive and negative direction. Besides, the total number of horses is 18002 in the testing dataset and the horses with 0 place difference occupies the most with a frequency of 2543 which is 14.126% of total.

Furthermore, the tables below show the accuracy of predicting the 1st placed horse (WIN) and top 3 placed horses (PLACE).

Table 4.2a Accuracy of WIN Prediction

Total 1st placed horses	1459
Correct Prediction	357
Percentage of accuracy	24.537%

Table 4.2b Accuracy of PLACE Prediction

Total top 3 placed horses	4409
Correct Prediction	2079
Percentage of accuracy	47.153%

To conclude, although the distribution shown in figure 4.16 is not satisfactory, by comparing to the accuracy of predicting WIN and PLACE betting to the previous final year projects, the overall result is acceptable.

4.4.7 Confidence Level of Prediction

Random Forest is constructed by grouping a number of decision trees and the finishing time results are predicted by the average of the outputs from all decision trees in the forest. However, it is difficult to evaluate the confidence of a time prediction by using general metrics, for instance, mean square error or explained variance since the prediction of all trees may actually not close to the reported finishing time.

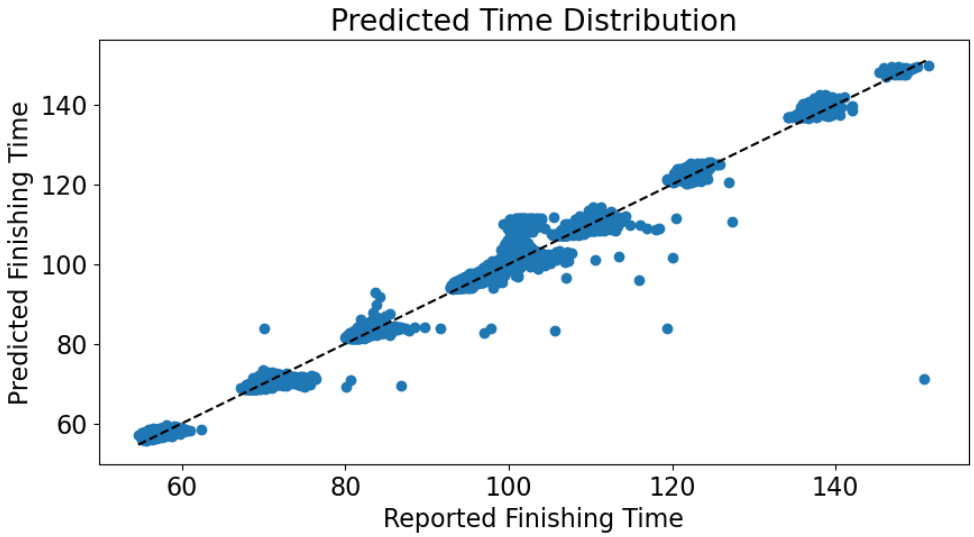

Figure 4.17 Distribution of Time Prediction

Figure 4.17 shows the distribution of the predictions from the random forest and the closer blue dots are to the dotted line, the higher accuracy of the prediction is. From the graph, different groups of datasets exist because of the distance difference between the races and the majority of predictions are close to the label. Since betting options need to be made based on the finishing time prediction, the accuracy of the prediction needs to be enhanced to boost the profits using the confidence level of the prediction.

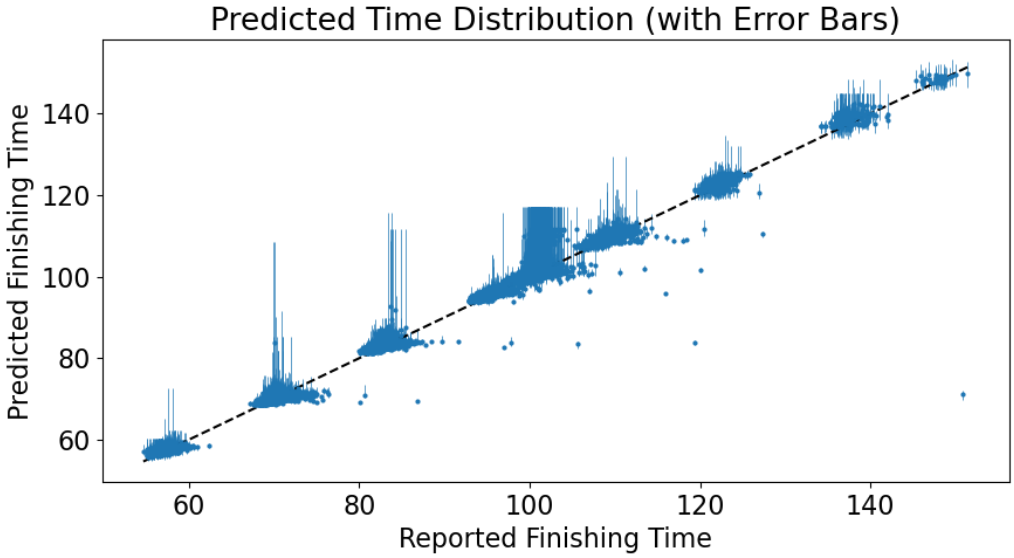

Figure 4.18 Distribution of Time Prediction (with Error Bars)

Figure 4.18 shows the same distribution as figure 4.17 while error bars are added indicating the range of predictions from different decision trees. There are some predictions with finishing time between 70 to 90 that have a relatively larger range of prediction results.

In order to determine the confidence level of the predictions from different decision trees, predictions of each testing data computed by each tree are recorded and check whether the corresponding labels lie between the 10-percentile and 90-percentile of the distribution of predicted results which indicates the confidence of the time predictions. [4][44]

- There are total of 42.884% of labels lie between the 10-percentile and 90-percentile of the prediction range.
- The maximum range of predictions is 38.518s and the minimum range of predictions is 0.30381s.
- The average range of predictions is 1.6429s and 26.480% of prediction range is greater than the average.

It is obvious that the result is not satisfying since more than half of the labels are not lying within 10-percentile and 90-percentile, which means betting on those options would be more likely to lose. Furthermore, the range of predictions varies to a large degree and thus, it is necessary to design a betting strategy that can avoid betting on those options that have a significant range of predictions greater than the average range.

Furthermore, the tables below show the accuracy of predicting the 1st placed horse (WIN) and top 3 placed horses (PLACE) excluding those predictions that have an error range larger than the mean range.

Table 4.3a Accuracy of WIN Prediction within Error Range

Total 1st placed horses	1459
Correct Prediction	357
Percentage of accuracy	24.537%

Table 4.3b Accuracy of PLACE Prediction within Error Range

Total top 3 placed horses	4409
Correct Prediction	1758
Percentage of accuracy	39.873%

Compared to the accuracy discussed in section 4.4.6, the accuracy of predicting 1st placed horse remains unchanged while the accuracy of predicting top 3 placed horses decreased by 7.28% after applying error range as an indicator. Although the results shown in this section look worse than

before, the betting strategy based on the confidence level has outperformed the other strategies which will be discussed in section 4.6.

4.5 Reduced Random Forest

The purpose of building the following new random forest model with fewer features is to improve the accuracy of the finishing time prediction result and the performance of the horse betting simulation. On the other hand, only the features with top feature importance in the previous model are used as input for the following model since the degree of influence of those features can be shown without affecting by other features that are not important. Thus, it would be more apparent to study the effects of features affecting the finishing time and the relationships between the features and results. For the evaluation of the reduced model, the same metrics and analysis would be used while the structure of the tree and features importance of the model would be discussed since the results and the interpretation of the results are similar.

4.5.1 Time Training Procedures

The training data contains 108107 records from season 2008/2009 to 2018/2019 while the testing data contains 18002 records of seasons 2019/2020 and 2020/2021. The size of training data and testing data remain unchanged while the number of features of the input data would be reduced rapidly.

Table 4.4 Features of Input Data

Features	**Types**	**Encoding Methods**
raceclass	Categorical	Ordinal
horseid	Categorical	Ordinal
distance	Categorical	Ordinal
rating	Real Value	/
winodds	Real Value	/
win_odds_rank	Real Value	/
raceidseason	Real Value	/

For the features in table 4.1, the features are selected based on both impurity-based and permutation-based feature importance and the encoding methods of the features are the same as in the previous training procedures.

4.5.2 Partial Dependence of Features

Compared to the previous models, the number of features is reduced and thus, the dependence of some would be increased or decreased and the significance of a feature would be shown clearly without affecting by other non-important features. In contrast, the majority of the partial dependence of selected features has no obvious changes.

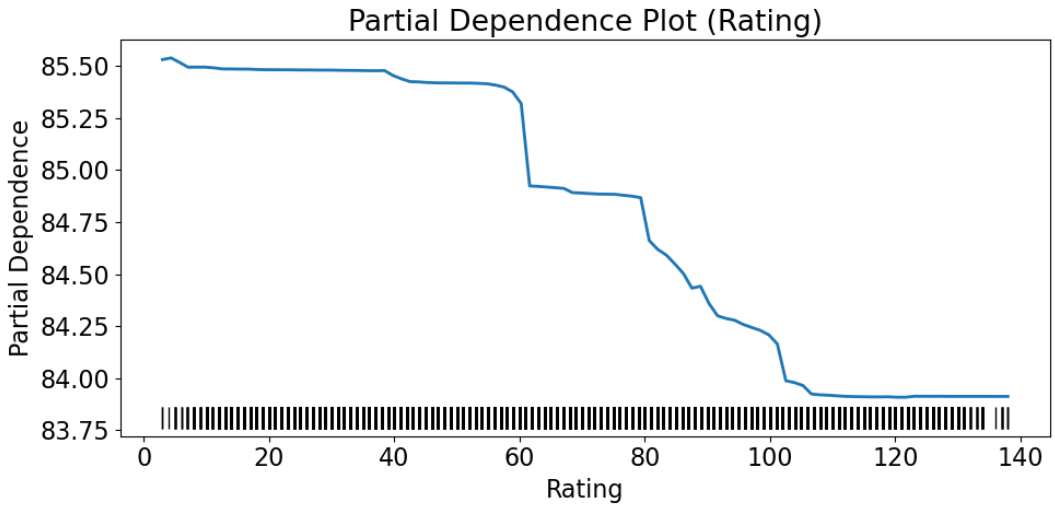

Figure 4.19 Partial Dependence Plot of Rating

Comparing figures 4.19 and 4.10, both of the partial dependence plots of rating have a decreasing trend with clear intervals and inversely proportional to the value of rating, while the values of the partial dependence are not the same and the range of partial dependence of rating in figure 4.19 is slightly larger than the one for the previous model. In the later section, we will show that the accuracy of the new model is better than the old model which implies that the impacts of rating in the reduced model is greater than in the previous model.

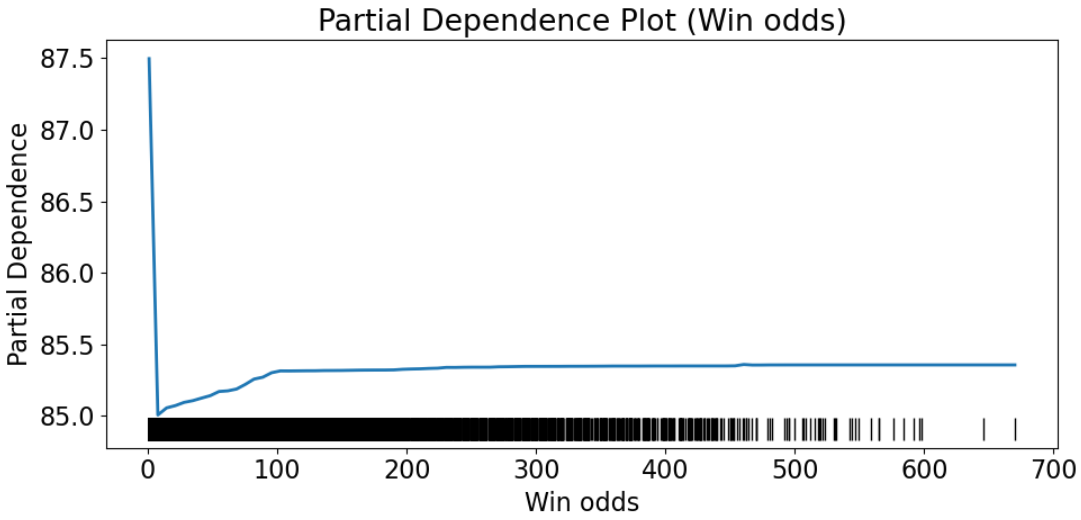

Figure 4.20 Partial Dependence Plot of Win Odds

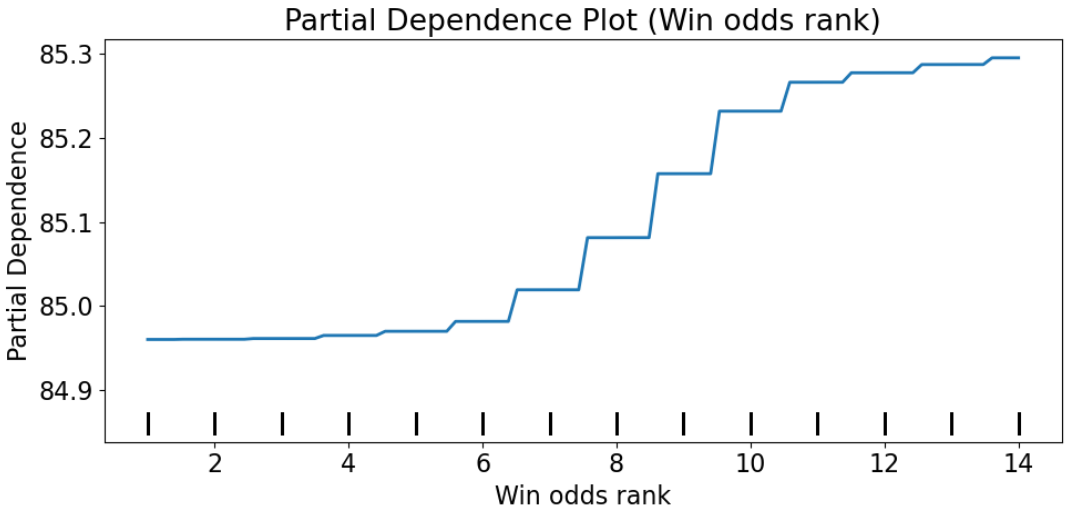

Figure 4.21 Partial Dependence Plot of Win Odds Rank

Figures 4.20 and 4.21 show the partial dependence relating to the odds value of horse in races and comparing them to the partial dependence from the previous model respectively (figures 4.11 and 4.12), there is not much difference between the graph and the range of the partial dependence which implies that the influence of win odds and the rank of win odds have been reflected already in the previous model.

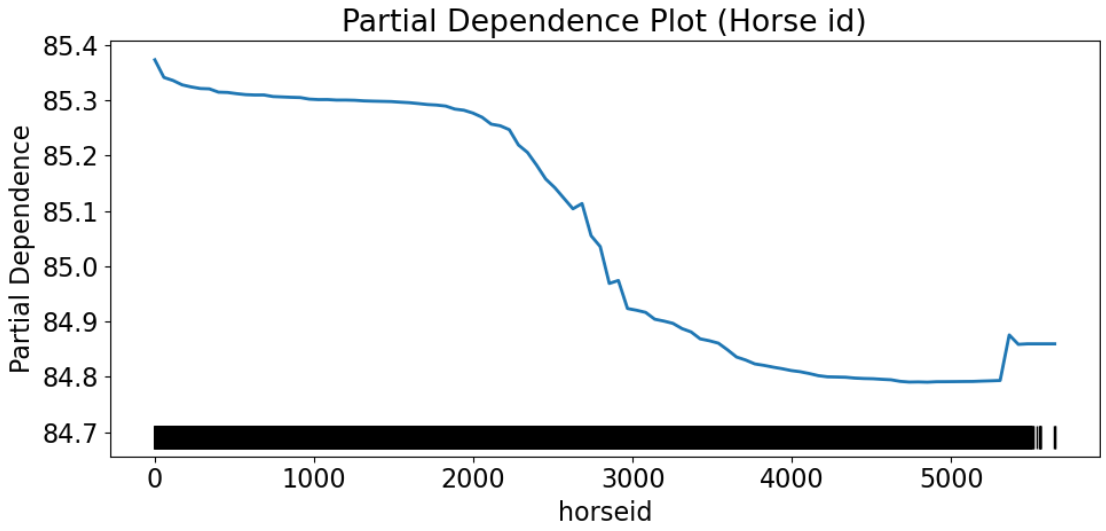

Figure 4.22 Partial Dependence Plot of Horse ID

Similar to the partial dependence of rating, the characteristics of the partial dependence of horse ID in previous and current models are similar. Overall, there is a decrease in partial dependence while the value of horse ID increases, and the decreasing rate is the highest within the range of horse ID values from 2000 to 4000. The range of partial dependence before horse ID value 1000 increases while the range of partial dependence after horse ID 5000 reduces which shows different values of horse ID has different changes in effects to the result of finishing time prediction.

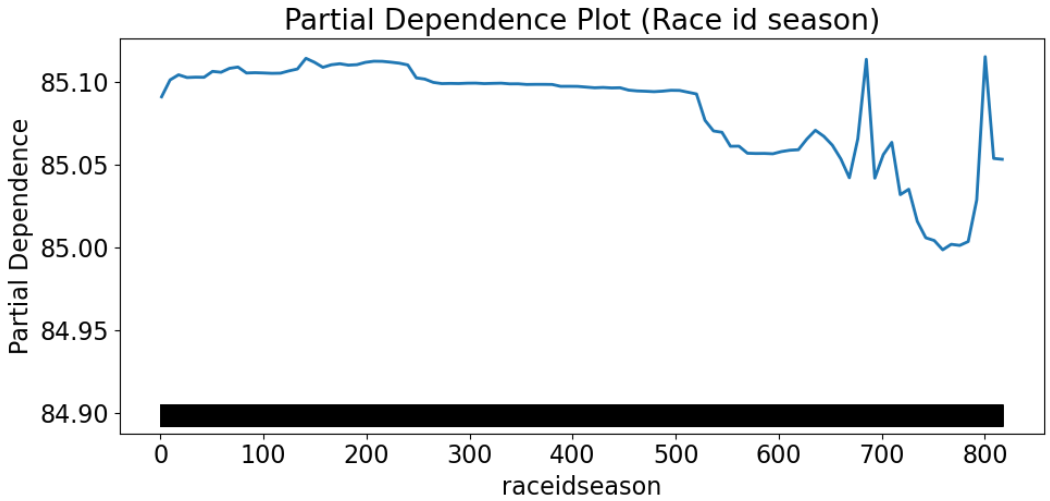

Figure 4.23 Partial Dependence Plot of Race ID Season

In figure 4.23, the curve shows the partial dependence of race ID season which indicates the index of a race in each season. The value of race index has no apparent meaning while it is correlated to

the with weather seasons, for instance, races indexed between 150 and 300 would be held in winter approximately in each race season. From the graph, fluctuation of partial dependence is shown which implies that index of races has influence on the finishing time prediction, while the range of partial dependence is relatively narrower than the others which means that the impact of the race index is not significant.

4.5.3 Evaluation Metrics

The mean squared error of the predicted and actual finishing time and the explained variance score of the random forest model would still be used as evaluation metrics for the reduced random forest model. Recall that the mean squared error of prediction results and actual finishing time shows the accuracy of the overall prediction and the explained variance score measure the discrepancy between the model and the data.

As mentioned in section 4.4.5, for the **previous model**, the following results are obtained

- The mean squared error of the predictions is 2.2649 seconds.
- The explained variance score of the model is 0.99388.

For the **reduced random forest model**, the results are shown below

- The mean squared error of the predictions is 1.7177 seconds.
- The explained variance score of the model is 0.99547.

For the mean squared error, the value has reduced by 24% with value of 0.5472 which shows that the overall accuracy of the finishing time prediction has enhanced. Recall that the common difference of finishing time between the first and the second horses are in the range of 0.1 to 0.3 seconds, the performance of the new model has significantly improved in terms of predicting the finishing time of horses while the overall performance of both models is still acceptable since the mean squared error is significantly greater than the common finishing time difference in real horse races.

On the other hand, the value of explained variance score has increased by 0.00159 while the scores of both models are close to 1. In terms of model training, both of the time prediction models are successfully trained, and the training data are well fitted into the random forest while the new model has only a slightly stronger association with the actual data which has a very limited impact on the prediction results.

4.5.4 Difference between Actual and Predicted Place

In order to simulate the horse betting strategy using the results of the finishing time prediction of the random forest, the predicted place of each horse in test data would be generated by ranking the

order of predicted finishing time in ascending order. The following figure shows the distribution of the difference between the actual and the predicted place ranked by finishing time prediction.

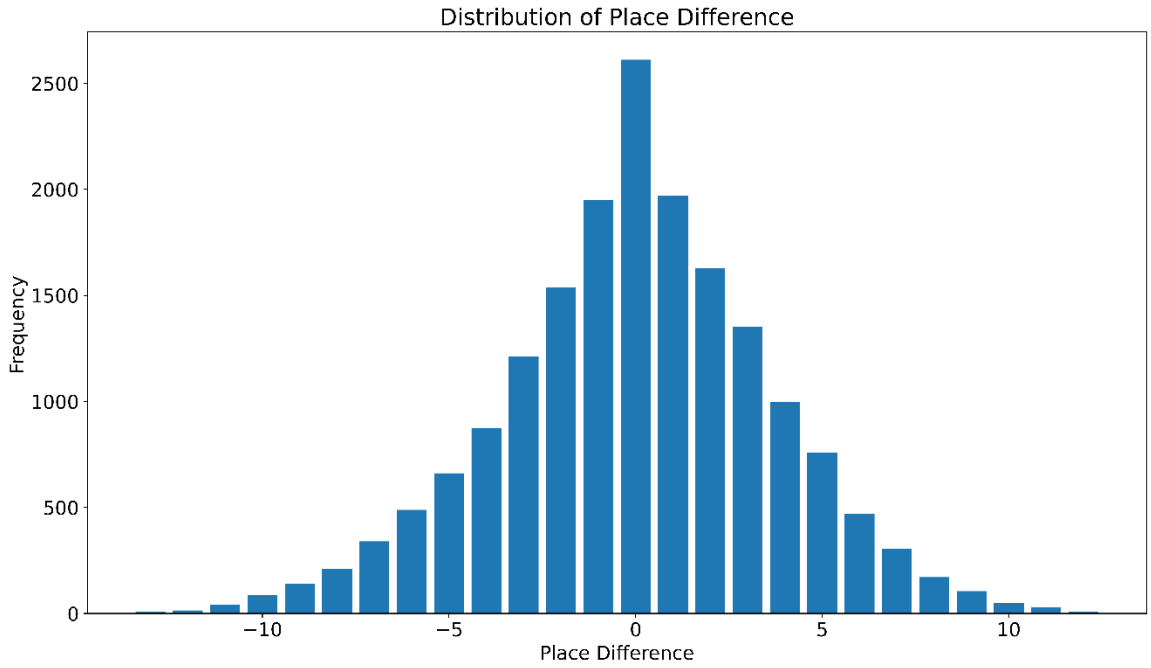

Figure 4.24 Distribution of Place Difference

The place difference is calculated by subtracting the actual place from the predicted place, and thus, both positive and negative numbers would be the result. The negative numbers mean the predicted place of the horse is overestimated and vice versa, and 0 means the prediction is correct. The distribution of place difference can be used to evaluate the accuracy of the finishing time prediction in terms of place prediction.

From the graph, the value of place difference is concentrated at 0 and the distribution decreases in both positive and negative direction overall which is identical to the results of previous model shown in section 4.4.6. Besides, the total number of horses is 18002 in the testing dataset, recall that the horses with 0 place difference occupies the most with a frequency of 2543 which is 14.126% of total in the previous model. For the new model the frequency has slightly increased by 83 to 2611 and occupied 14.504% of total results which shows that the new model has no significant improvement in the accuracy of the predicted place.

Furthermore, the tables below show the accuracy of predicting the 1st placed horse (WIN) and top 3 placed horses (PLACE).

Table 4.5a Accuracy of WIN Prediction

Total 1st placed horses	1459
Correct Prediction	355
Percentage of accuracy	24.331%

Table 4.5b Accuracy of PLACE Prediction

Total top 3 placed horses	4409
Correct Prediction	2077
Percentage of accuracy	47.108%

Compared to the previous results, the accuracy of WIN prediction of the new model has slightly reduced by 0.206% and the number of correct predictions for PLACE has slightly reduced by 20 to 2077 and the accuracy percentage has decreased by 0.499%. Overall, the performance of horse betting has a similar profit compared to the previous model.

4.5.5 Confidence Level of Prediction

Recall that the purpose of finding the confidence level of the finishing time prediction is to help improve the horse betting simulation in later sections since the level of confidence of the results is able to act as an indicator which helps deciding betting options and this cannot be substituted by other general metrics such as mean squared error or the explained variance score.

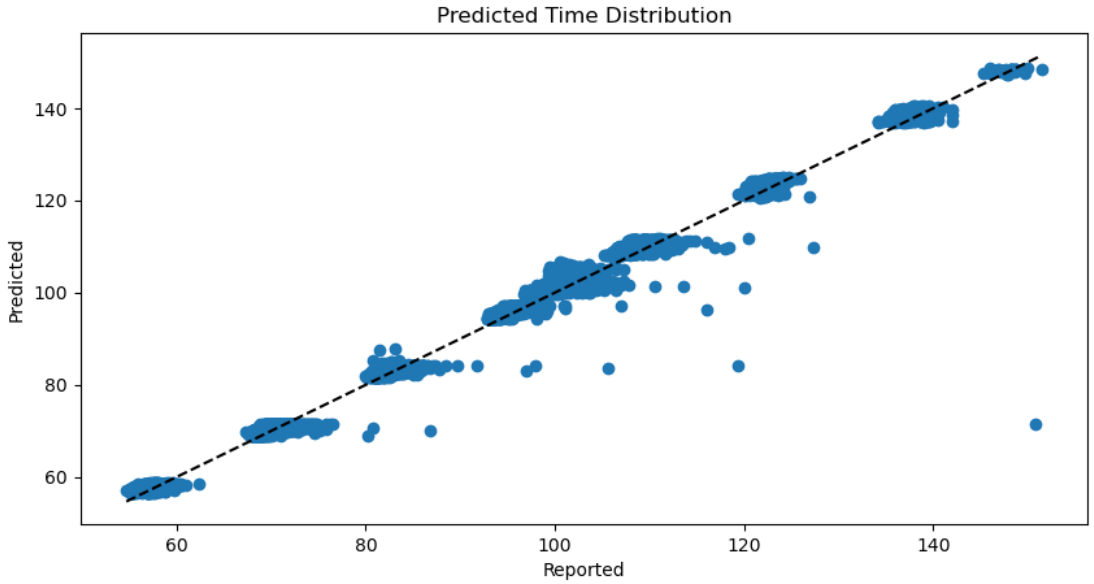

Figure 4.25 Distribution of Time Prediction

Figure 4.25 shows the distribution of the predictions from the random forest and the closer blue dots are to the dotted line, the higher accuracy of the prediction is. Similar to the graph in section 4.4.7, different groups of datasets exist due to the different distance races. The following graph would provide a more detailed view of the performance.

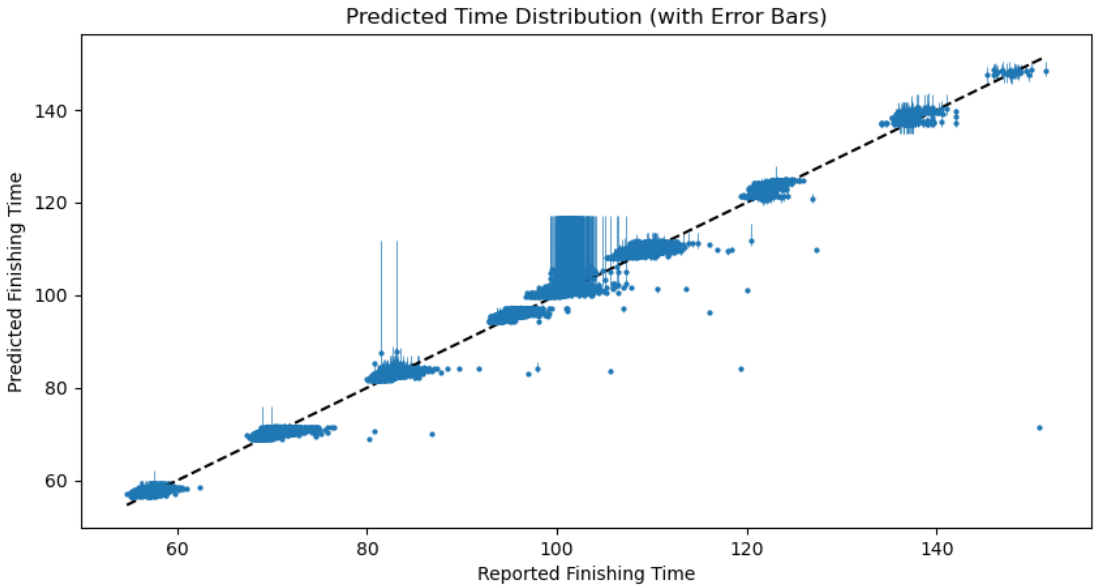

Figure 4.26 Distribution of Time Prediction (with Error Bars)

Figure 4.26 shows the same distribution as figure 4.25 while error bars are added indicating the range of predictions from different decision trees. Comparing to figure 4.18 for the previous mode, similarly there are some predictions with finishing time around 100 have a similar range of prediction results, while the number of error bars and length of error bars are reduced excluding the predictions around finishing time value 100, which shows that the results of different trees in random forest have a less discrete result relatively.

Recall that the confidence level of predictions is determined by checking whether the actual finishing time of the test data lies between the 10-percentile and 90-percentile of the predicted results.

As mentioned in section 4.4.7, for the **previous model**, the following results are obtained

- There are total of 42.884% of labels lie between the 10-percentile and 90-percentile of the prediction range.
- The maximum range of predictions is 38.518s and the minimum range of predictions is 0.30381s.
- The average range of predictions is 1.6429s and 26.480% of prediction range is greater than the average.

For the **reduced random forest model**, the results are shown below

- There are total of 22.497% of labels lie between the 10-percentile and 90-percentile of the prediction range.
- The maximum range of predictions is 28.607s and the minimum range of predictions is 0.19986s.
- The average range of predictions is 0.85121s and 22.397% of prediction range is greater than the average.

It is obvious that the result is not satisfying and is worse than the previous model by dropping the number of labels lying between the 10-percentile and 90-percentile of the prediction range significantly. The percentage of labels lying between the range has decreased by 20.387% and one of the reasons could be the range of prediction from different trees in the random forest has greatly reduced. The maximum and minimum range of predictions have both dropped by 25.730% and 34.215% respectively which increases the probability of the labels lying outside the range. Besides, the mean range of predictions has decreased by 48.188% which shows the trees in reduced random forest have a more concentrated result than before.

Furthermore, the tables below show the accuracy of predicting the 1st placed horse (WIN) and top 3 placed horses (PLACE).

Table 4.6a Accuracy of WIN Prediction within Error Range

Total 1st placed horses	1459
Correct Prediction	355
Percentage of accuracy	24.331%

Table 4.6b Accuracy of PLACE Prediction within Error Range

Total top 3 placed horses	4409
Correct Prediction	1848
Percentage of accuracy	41.914%

Compared to the previous results, the accuracy of WIN prediction of both the previous model and the new model are identical while the number of correct predictions for PLACE has slightly increased by 90 to 1848 and the accuracy percentage has increased by 2.041%. Although the accuracy of predicting the top 3 placed horse based on the error range is slightly greater than the previous model, simulation in real betting in section 4.6 will shows that the profit of betting based on table 4.6b is worse than the previous model.

4.6 Simulation on Betting

As the predicted places of each horse are computed from the finishing time prediction, a simulation of wagering both WIN and PLACE based on the predicted place can be used to evaluate the performance of the model. Assuming that $10 would be used to bet on the horse that is predicted to be the winner for each race. If the horse bet is 1st placed in that race, a profit of $10 * odds - 10$ would be added to the cash balance, otherwise, $10 would be deducted from the cash balance.

4.6.1 Betting WIN

In order to investigate if the time prediction model is able to make a certain amount of profit stably, the following figure shows the cash balance of betting WIN according to the predicted place.

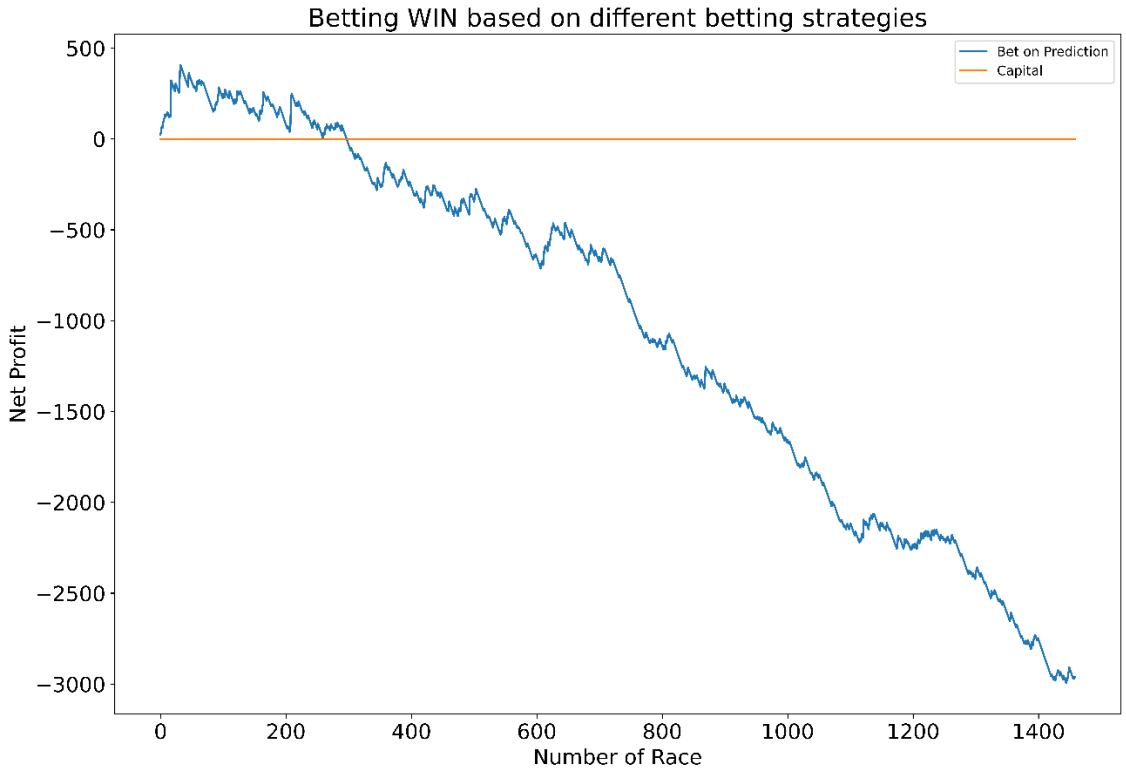

Figure 4.27 WIN Betting Simulation

From the graph, the cash balance decreases continuously and the is a net loss of $2960 after simulating the WIN betting based on the result from the prediction. There are about a quarter of predictions which is correct while the profit earned is relatively less than the loss in wrong betting. Thus, the net flow of money would decrease continuously.

In terms of profit, a comparison can be made between betting according to predictions and different betting strategies, for instance, betting on the lowest odds or lowest rating etc., and thus, the performance of prediction can be measured. The graph below demonstrates the cash balance of betting WIN with different betting strategies and the calculation of the cash balance is the same as above.

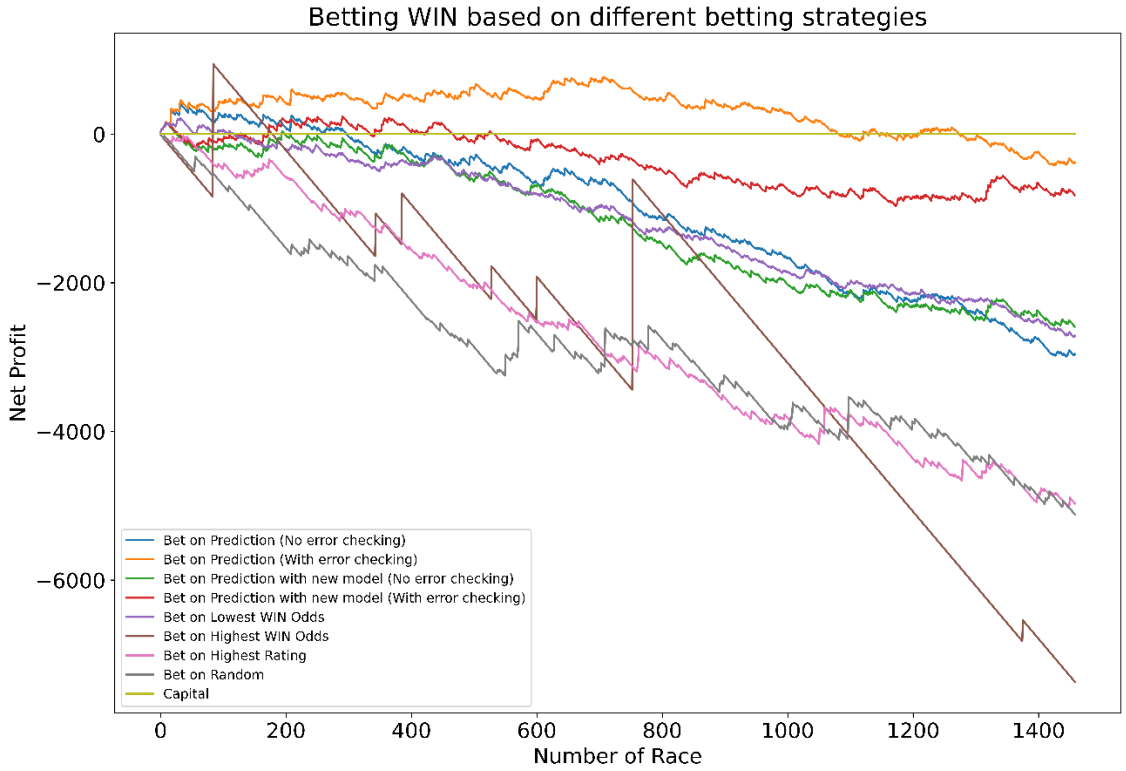

Figure 4.28 WIN Betting Simulation with Different Strategies

In the graph, 3 different betting strategies are compared with betting WIN based on the prediction. The strategy which has the worst result is betting on the highest WIN odds in each race which has a total loss of $7370 while in the middle of the simulation, it gains a huge amount of profit and has a net profit higher than some strategies at that moment.

Betting on random refers to picking a horse in each race randomly without any consideration or comparing and with an average probability of betting correctly around 7% which has a similar total profit as betting according to the rating of horse in each race.

From the graph, although betting randomly recorded the lowest net balance at beginning, the performance of it is just slightly better than the strategy of betting on rating, and the curve has the greatest fluctuation which shows the stability of this strategy is relatively low. Besides, the result of betting on prediction including both old and new model without error range, and lowest odds is

close most of the time while betting on prediction from reduced random forest has a better performance in the end.

On the other hand, betting on prediction based on error checking refers to betting options would be made only when the range of predictions from different trees in random forest is less than the mean, otherwise betting options would not be made, and no money is spent. From the graph, betting on model prediction and based on the error checking have the best performance no matter which models the results are from and the performance of betting on the old model is better than the new model while the difference of profit is narrowing at the end.

Although all betting strategies are not able to generate profits, making betting options based on error checking can help reduce the number of bets as well as reducing the number of losses as mentioned in chapter 4, the accuracy of WIN betting remained unchanged after applying error range checking. It implies that if simply betting on all races with the predictions from the random forest has a similar result as betting according to the lowest WIN odds in each race which regards as public intelligence mentioned in section 3.3.3.

4.6.2 Betting PLACE

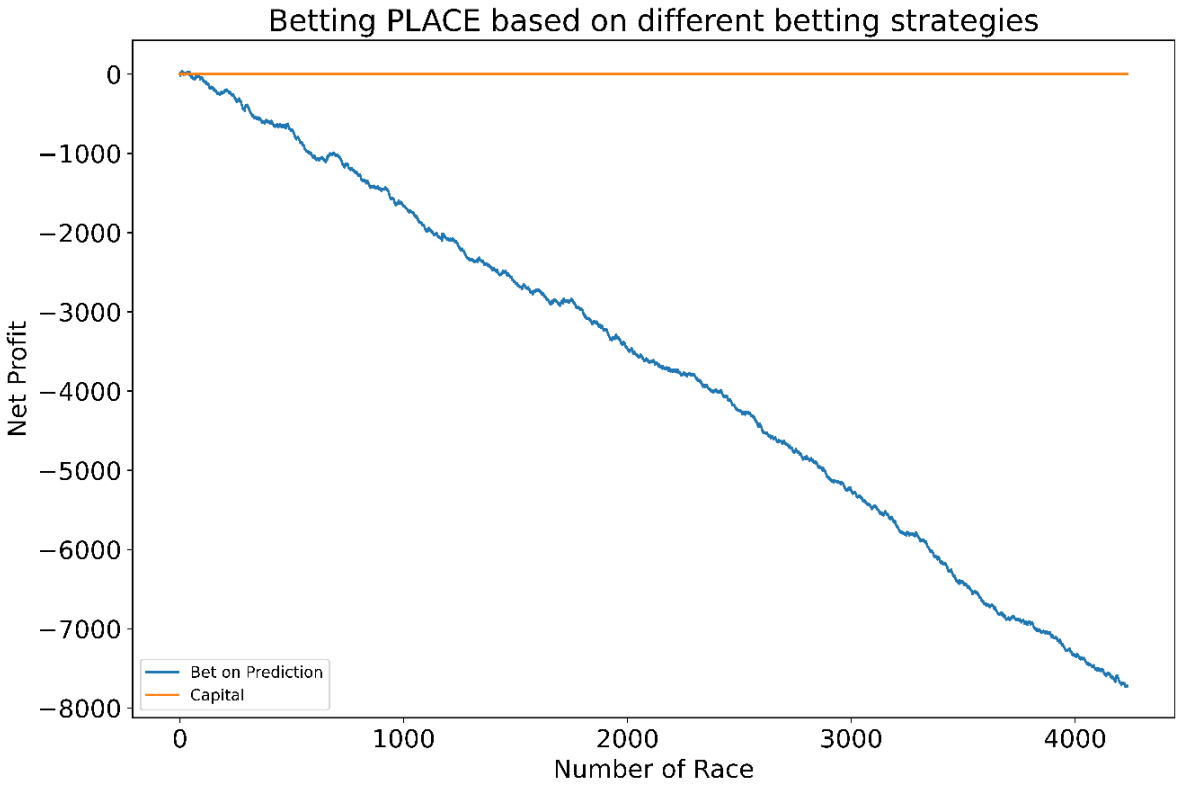

Figure 4.29 PLACE Betting Simulation

The approach to bet PLACE is betting on all horses which are predicted to be the first 3 placed in the race, and the calculation is same as above. Figure 4.29 shows the cash balance of betting PLACE according to the predicted place.

From the graph, the cash balance decreases continuously and the is a net loss of $7727 after simulating the PLACE betting based on the result from the prediction. However, as mentioned in section 4.4.5, the accuracy of PLACE prediction is 47.153%. There are about half of predictions which are in the range of top 3 while the profit earned is much less than the loss in wrong betting. Thus, the net flow of money would decrease continuously.

The graph below demonstrates the cash balance of betting PLACE with different betting strategies and the calculation of the cash balance is the same as above.

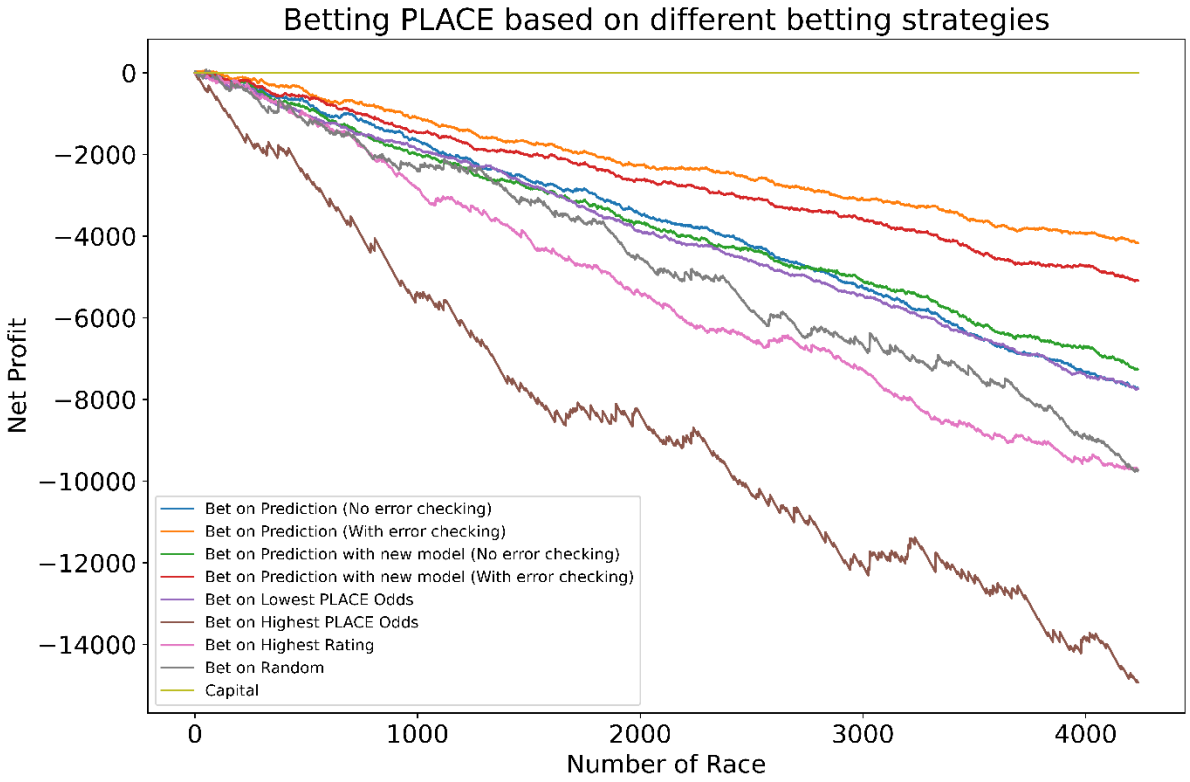

Figure 4.30 PLACE Betting Simulation with Different Strategies

Similar to figure 4.28, the strategy which has the worst result is betting according to the highest PLACE odd in each race which has a total loss of $14931. Betting on a random horse in each race randomly without any consideration or comparing and with an average probability of picking one horses which is in top 3 place is around 21% to 25%, and it has a better performance than betting on the horse with the highest rating in each race majority of the time while final cash balance is close to each other.

Besides, the result of betting on predictions of the old model and lowest odds are close from the beginning while the gap between them starts expanding from the middle, which shows the net profit of betting based on odds has a slower decreasing rate than the prediction one. The performance of betting based on the predicted place of new model is close to the results of old model and lowest odds for most of the time, while it has a higher cash balance at the end.

On the other hand, similar to betting on WIN option, betting on model prediction and based on the error checking have the best performance no matter which models the results are from and the performance of betting on the old model is better than the new model while comparing to the performance of betting on WIN, the cash balance for betting PLACE is much less than betting WIN and the trend on both curve decrease most of the time without any significant uptrend. Besides, the cash balance of old and new models does not tend to converge at the end which shows the performance betting PLACE based on the old model with error checking is generally better than the new model.

The greatest difference of betting WIN and PLACE is that the range and difference of odds of each horse are distinct. For instance, the final WIN odd of the most unpopular can be more than 1000 while the greatest final PLACE odd in our record is only 121. Although the accuracy of predicting top 3 horse has almost 50% which is satisfactory while the money loss is much greater than the money gained. Overall, the simulation shows that the profitability of gambling based on the prediction with error checking does outperform the public intelligence while still no profit can be generated.

4.7 Summary

In summary, the procedure of time prediction model training is successful since the decision tree is validated to split the data based on the features that determine the finishing time by inspecting the structure of a decision and the decision path. Furthermore, the training data is well fitted into the random forest model which is verified by the high explained variance score.

In addition, by inspecting the importance of features and the decision path, the prediction made by the model is complete and all-around, since different types of features, for instance, environmental (going, racetrack), horse related (rating, past performance) and public intelligence related (WIN odds) are taken into consideration.

On the other hand, a new random forest model is trained, and the size of input features is reduced which is selected based on the features importance computed with the old model. The new random forest model has shown the influence of different features with the partial dependency plot.

In terms of accuracy, the predicted finishing time for each horse is not satisfactory while the accuracy of predicting WIN and PLACE achieved an acceptable result by comparing to the

previous projects. In terms of profitability of the model, a new strategy of avoid betting on the predictions, which have a greater variance of predictions by different trees in the forest, has greatly improved the profitability compared to the result in the last semester while none of them are able to return any profit.

5. Horse Betting in Reinforcement Learning

5.1 Multi-armed Bandit Algorithm

5.1.1 Introduction

The multi-armed bandit (MAB) problem is a classical problem that demonstrates the **explore-exploit dilemma**. The name "bandit" comes from the imaginary scenario where one is playing slot machines (just like bandit that robs money) in a casino. Each slot machine will have different rewards, some give more rewards on average than others. You can only play one slot machine at a time and the goal is to earn more money in fewer total rounds played or under limited budget. Either you can play the one that gives the best reward (exploit) so far or play others that can potentially give better reward (explore). Similar scenario can be commonly seen in daily lives. Examples are advertisement selection [48], article recommendation [19], portfolio allocation [28], where MAB algorithms are actually applied. For the horse betting problem, it has similar settings where there are also multiple options (horses) to choose from, and their reward are uncertain. Exploit can mean choosing those that have higher chance to win but earn little (lower betting odd), explore can mean choosing those that less likely to win but may potentially earn more (higher betting odds). Note that there is some difference from the common settings though, which is the rewards of all actions can be seen afterwards. This is called full-feedback setting while the setting that bandit algorithms usually deal with is bandit setting where only reward of chosen action(s) can be seen. The former is less restrictive than the latter. However, we can still use the latter setting and attempt to apply bandit algorithms.

5.1.2 Modeling Horse Betting as MAB Problem

There are two major variants of formulation of the MAB problem. One is **stochastic** bandit**;** another is **adversarial** bandit. The former assumes there are fixed but unknown reward distributions for each action. The latter doesn't assume any reward to follow certain distribution but rather determined by an adversary that could even possibly setting rewards against actions we would take. This can be more general than the former but more pessimistic. We will focus on the former as there are more available algorithms that would better support the contextual setting that would later be discussed, and we believe there could be some distributions for the rewards.

5.1.2.1 Formal Definition of Stochastic MAB Problem

It is a sequential game with total of T rounds and all the possible arms (actions) form an arm set A. At each round t, the available arm set is $A_t \in A$. The player plays an arm $a_t \in A_t$ and observe reward r_{t,a_t} is some random variable $X_{a_t,}$ drawn from a distribution $p_{a_t,}$ of corresponding arm. Available arm set, and corresponding reward of each round are independent from each other. Usually the goal is to maximize expected cumulative reward $\mathrm{E}\left[\sum_{t=1}^{T} X_{a_t}\right]$ or equivalently minimize the cumulative regret defined as $\sum_{t=1}^{T} \max_{a \in A_t} E[X_a] - E\left[\sum_{t=1}^{T} X_{a_t}\right]$ which is to compete with the best policy which always pick the action with highest expected reward (in our case, optimal policy is always betting on the horses that would win the race). For horse betting problem, the arm set can be each particular horse or horses in a race which are given certain order (e.g., by their winning chance) or other possibilities, this will be discussed in later chapters.

For stochastic bandit, two representative algorithms are upper-confidence-bound (UCB) algorithm and Thompson sampling algorithm. They have very different ways of estimating the expected reward and address explore-exploit dilemma. The former simply estimates the mean reward while the latter assumes some probabilistic distribution about the reward and try to update its belief about the distribution in Bayesian fashion. We will also use the variants of them in our experiment and see compare how they perform.

5.1.2.2 Contextual MAB

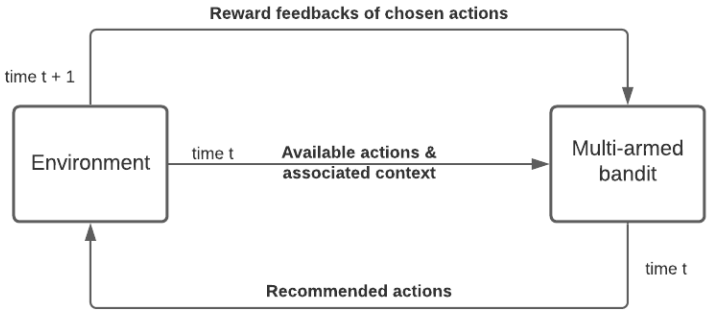

Figure 5.1 Flow Diagram of Contextual MAB

In a standard MAB problem as what is described in previous part, the only information the player can get about the action is only the rewards received. While in reality, there can be more side information. For example, in article recommendation, say each action is an article, we can know about things such as its genre and popularity, and we also have the information about the user such as his preference. This information could aid us in estimating the reward such as whether this

article would be viewed by the user. Similarly, in horse betting, we also get access to information like the changing betting odds for that horse, the winning chances of it, etc. We can possibly make good use of this information to predict the reward. So formally, in each round t, for each action a_t, we can know its context x_{t,a_t}which might have some relation with reward r_{t,a_t} . Some examples like LinUCB [36], it assumes its context to have linear relation with the expected reward. That is, $\mathrm{E}[r_{t,a_t} | x_{t,a_t}] = {x_{t,a_t}}^T \theta^*_{a_t}$, where coefficient $\theta^*_{a_t}$ is specific to action a_t. This is a simple model but worked well in Internet advertisement selection [48]. Whether this is applicable to our problem is unknown, but we will try to use algorithms with similar idea in later parts as some initial trials for its simplicity.

5.1.2.3 Possible Models for Horse Racing Betting

If we try to model horse betting as a MAB problem, one important thing is how do we define the arm (action) set. There are two possible basic approaches we can think of.

Horses in One Race as Action Set

For a horse race, there are some ways we can tell people's confidence on winning chances of each horse. For example, by looking at the betting odds, the horses that are bet more have lower odds, and those that are bet less will have higher odds. So, we can generally group horses by ranking and for a 14-horses race, there can be 14 actions. 1st action to be the one most likely to win, 2nd to be second likely to win, and so on. And in our experiment, we would use the finishing time predictions of the random forest model for ranking.

Advantage

- We can have enough samples to go through.

Disadvantage

- Either we use betting odds or our own predictions as reference to the ranking, it's nowhere to be totally accurate.
- The ranking is absolute and doesn't reflect the actual difference which can be minor. In fact, reordering of the top places are commonly observed.

All Horses as Action Set

We can include all horses in an arm set. In each race, there is only a small subset of the arm set that is available and allowed to be played (bet on).

Advantage

- As each horse should behave somewhat differently, if we maintain a specific set of parameters about the reward estimation for each horse, it could be more accurate.

Disadvantage

- Some horses can participate up to 30 races, some horses just appear as few as just 1 race. There is no way we can estimate their performance well. Although there are some approaches to maintain some general parameters applicable to all actions, the more advanced ones are complicated, and we can't have enough time to well explore these approaches in this semester. However, we will still try to attempt some of these approaches that are simpler but might not work well.

We will try both approaches. And we will include "not bet" as an action in the two approaches as well. This is also for filling up the races with less than 14 horses.

5.1.2.4 The types of Bets to play

There are many types of bets in HK horse racing. In the past years of FYPs, they focus on two betting types which are win bet and place bet. One can earn from win bets only when they bet on the winning horse. This may be effective in evaluating prediction accuracy but not particularly meaningful to use any betting strategy on it. Therefore, we will focus on place bets which will give more flexibility as getting any out of top 3 horses right is fine.

5.1.2.5 Combinatorial MAB

Since for place bet, bet on any of top 3 horses will make us earn (not by much). We allow the agent to play multiple actions in one round. Such setting that allows multiple action to be played is known as combinatorial MAB. Formally, if there are m arms (also called as base arms) in one round, these base arms can form super arms, pulling super arms mean pulling all the base arms. The number of base arms to form the super arm can be unlimited but we can also have a budget B that limits the maximum number of arms that can be chosen. The rewards can simply be the total sum of rewards of all base arms in our case (while there are some that would consider diminishing return [14]). We can use standard MAB algorithm to solve this problem but there are some algorithms specialized for these types of problems and we will also try that.

5.1.2.6 Reward Functions

There are two natural ways of defining the reward for each arm.

Bet Right or Wrong

In many of the scenarios where bandit algorithms are deployed, rewards are bounded between [0,1], and theoretical guarantees are often built on this assumption. Therefore, it might be sensible to do similar. We can define the reward as follows.

- $R(Bet\ any\ of\ top\ 3\ horses\ correctly) = 1$
- $R(Bet\ wrong) = 0$

It's trickier to define the reward for not betting

We wouldn't set it to 0 because it will not be better than any option. And we don't want the agent to always choose it over other options. Therefore, we can set it to be the place accuracy of our finishing time prediction, which is 46%.

- $R(Not\ bet) = 0.46$

Problem of this approach:

This approach disregards actual money gain, those that have lower win rate can have good overall earn, but they will likely to be ignored in this way. So, this approach might only be able to play safe on the one most likely to win.

Actual Cash Change

A more realistic approach is to really define reward based on dollar lost or gain.

- $R(Bet\ any\ of\ top\ 3\ horses\ correctly) = dollar\ bet\ \times\ odd\ of\ betted\ horse$
- $R(Bet\ wrong) = -dollar\ bet$

Dollar bet is chosen to be \$10 as it is the minimum amount of money to bet.

For reward for not bet, we set it to be slightly higher than the average amount of money change when betting randomly, which is -3.2

- $R(Not\ bet) = -3$

Problem of this approach:

As mentioned in the previous part, it would be better to bound the reward between [0,1]. If we directly use the actual dollar loss/gain, the variance will be very high and it's unknown whether the algorithms would apply.

Normalized Actual Cash Change

There are also ways to bound the reward function in 4.2.3.2 between $[0,1]$. We can consider remaining money (always greater or equal to 0) and divide it with some large number N to make it less than 1.

- $R(Bet\ any\ of\ top\ 3\ horses\ correctly) = dollar\ bet \times odd\ of\ betted\ horse/N$
- $R(Bet\ wrong) = 0$
- $R(Not\ bet) = dollar\ bet - 3 = 7$

N here is chosen to be 1300 as among those top 3 horses in our dataset, the highest odd is 121 and the dollar we bet each time is \$10.

Problem of this approach:

After division, the difference between rewards might be minor, to the point where it doesn't differ much.

5.1.2.7 Features to Use (Context)

- Last moment place odds (It is close to the final odds which can roughly tell how much we will earn if we bet right)
- Last 10 minutes exponential moving average (we can tell whether the odds are decreasing/increasing which reflects confidence in this horse),
- Rankings
  - Initial odds provided by HKJC, which reflects their predictions
  - Last moment odds before betting ends, this reflects confidence of gamblers, we call it public intelligence
  - Predicted finishing time from our random forest model

  Since bandit algorithms consider each action to be independent and don't consider the correlation between actions (which is not true in our case as the horses are competing with each other), hopefully adding information about ranking can help addressing the problem.
- The ratio between finishing time of each horse with that of the horse predicted to ranked 1 place ahead of it in ranking of predicted finishing time, which is to capture the relative difference between each place.

5.1.2.8 Overall Flow to Use Bandit Algorithms

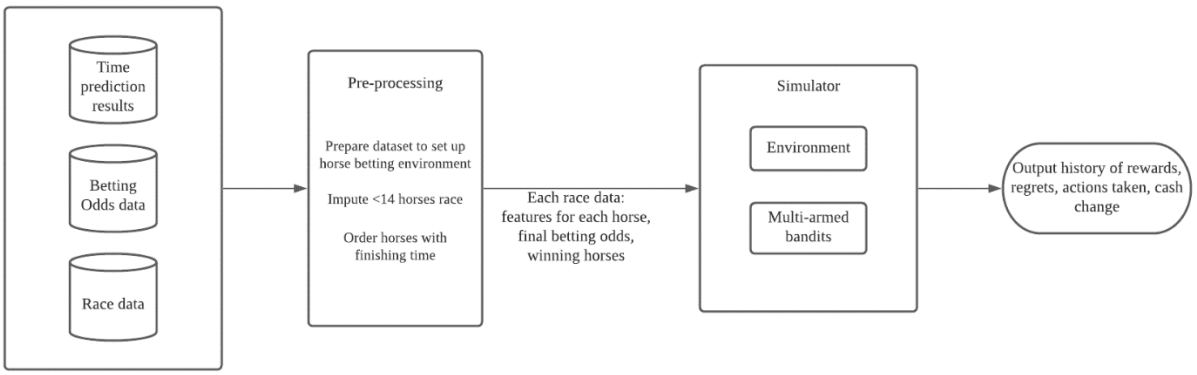

Figure 5.2 Overall Flow Diagram

Multi-armed bandit algorithms are essentially a kind of reinforcement learning that learns while playing inside an environment. In our project, we create a horse betting environment and simulate each race using historical data. And the agents will receive relevant context of each horse in each

race, and choose to bet on some horses, then will get reward feedback that is based on the result of the actual race. Then, finally we can see how the algorithms behave from the history of actions, rewards, and change in cash balance.

5.1.3 Algorithms to Use and Why

The mechanics of the following algorithms are already described in chapter 2. Here we would just restate why we would use these algorithms.

5.1.3.1 Contextual Stochastic Bandit

As stated in 4.2.1, we would like to use variants of two representative classes of stochastic bandit algorithms, upper-confidence-bound and Thompson sampling. And we will focus on algorithms such as LinUCB and LinTS that work on the simple assumption that context is linearly correlated with reward since it's easier to implement.

Standard bandit algorithms are only allowed to pick one action at a time. To handle the combinatorial scenario, usual way is to just pick the top k actions with highest scores given by the algorithms. We will give a prefix "k-" to the name of the algorithms to denote that k actions are picked.

Hybrid LinUCB

LinUCB has two versions. One is disjoint and another is hybrid. The former has only arm specific parameters for reward estimation. The later has shared parameters as well.

Therefore, we would use the latter and hopefully capture the correlation between arms.

Linear Thompson Sampling (LinTS)

We would like to use an approach parallel to UCB and Thompson Sampling is a common approach besides UCB.

5.1.3.2 Contextual Combinatorial Bandit

Betting on multiple horses (actions) falls in the realm of combinatorial bandit, therefore we would like to use some algorithms of that class to handle this problem.

CC-MAB

In each round, it uses at least some of its action budget for exploring actions that are less chosen while playing the actions that are more chosen and more likely to give higher reward. This is a more standard approach for solving the combinatorial bandit problem.

It also considers volatile arm set (i.e., the available arms are changing, this is the case of the action set formulation in 4.2.3.2.) by grouping actions with similar context and assume that they have similar estimated reward. It can be an experiment on the ways to define action sets.

5.1.4 Result and Analysis

5.1.4.1 Procedures

We will have two parts of experiment. In the first part, we do direct betting, which is taking whatever actions the bandit algorithms decide to take. In second part, we will make improvements on part 1 like using another bandit algorithms to decide whether to take the action or not or even bet more.

Metrics

Like what have mentioned in chapter 4.2, the usual metrics for evaluating bandit algorithms are cumulative reward and/or regret, which are equivalent. We will use regret, but our objective is different. For regret, usually the goal is to get regret go as sublinear as possible (i.e., the regret grows slower and slower overtime). However, optimal strategy in our problem is to always bet on the correct horses, which is too harsh, we will expect the agent to at least grow slow overall but not necessarily continue decreasing. We will also evaluate the change in cash balance.

Ways of Comparison

We will compare the 3 algorithms, random bet, public intelligence (betting on lowest odds), our predictions (bet on best predicted horses). For maximum horses to bet (budget B), we would set $B = 2$. For $B = 2$, as there are only 3 horses that can win, placing 3 bets is very possible to lose at least 1. Therefore, placing only 2 might be safer. For CC-MAB and LinTS, they will be run for 5 times to compute an average as they are randomized algorithms.

Data to Use

We have 1414 races in total, each with 5-14 horses, for those that are not up to 14 horses, we will append with records with horseid as -1 and all other features as 0 to make the dataset to have unified dimensions. In our programs, those with horseid -1 will be ignored. We will run it sequentially without replacement. We don't separate training set and testing set because the usual scenario where bandit algorithm is applied is online learning. This will pose also extra challenge as the algorithm just learns on the go, which will give it disadvantage at start.

Challenges of Horse Betting

For place bet, accuracy of public intelligence is 50%. And HKJC will take away 17.5% of money bet, and only then the remaining 82.5% will be distributed to the winners. This makes betting odd

often low. The below figure shows the betting odds and corresponding frequency for horses with lowest 3 odds (most likely to be in top 3).

```
{1.6: 444, 1.5: 433, 1.7: 409, 1.4: 397, 1.8: 392, 1.9: 333, 1.3: 317, 2.0: 286, 1.2: 249, 2.1: 210,
2.2: 174, 2.3: 128, 1.1: 127, 2.4: 79, 2.5: 72, 1.0: 59, 2.6: 50, 2.7: 34, 2.8: 20, 2.9: 7, 3.1: 5,
3.0: 4, 3.2: 4, 3.4: 2, 3.3: 1, 3.6: 1, 4.5: 1, 4.6: 1, 5.2: 1, 8.0: 1, 10.0: 1}
```

Figure 5.3 Betting odd frequency in our dataset

It can be seen that the betting odds always would be lower than 2 (for $10 dollar bet, getting back lower than $20 dollar). If place bet accuracy is 50% (losing half the time), then one has to earn at least $20 in one game to compensate the loss of $10 dollar in another game). In that way, people always keep on losing (which can also be seen in the graphs shown in the result). This makes it very hard to earn money.

5.1.4.2 Part 1: Direct Betting

Action set 1 and 2 refers to “Horses in a race as action set” and “All horses as action set” in chapter 4.2.3. Reward function 1 ,2 and 3 refers to “Win or lose”, “Actual cash change” and “Actual cash change but normalized” in chapter 4.2.5.

Action Set 1

- **Reward function 1**

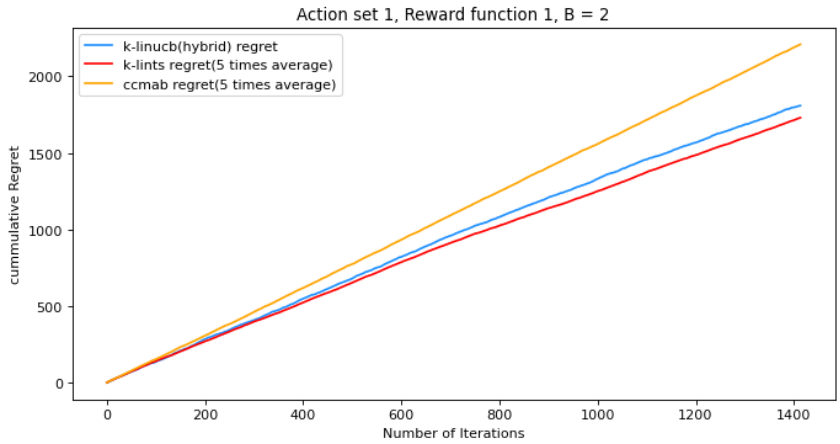

Figure 5.4 Cumulative regret for action set 1, reward function 1

We can see that the regret for LinUCB and LinTS are better than CC-MAB and show slight improvement over time. However, as previously mentioned, it’s not likely to continue reducing the regret as optimal policy is defined as guessing top horses correctly.

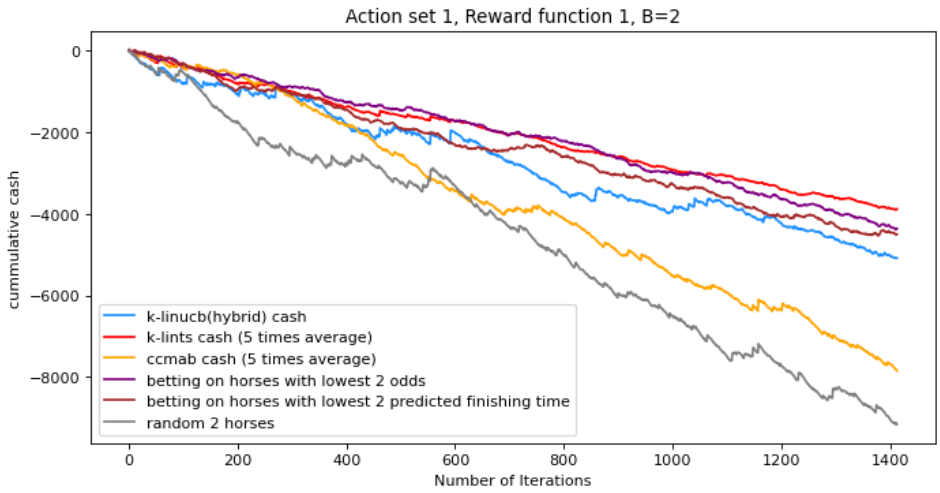

Figure 5.5 Cumulative cash balance change for action set 1, reward function 1

For all algorithms, they all lose money eventually. The best performing algorithm is only slightly better than betting on lowest odds/predicted time. This is reasonable as reward function 1 favors those horses with higher winning rate and the algorithm will tend to choose those more likely to win but also earn less (very low betting odd). And we can also see that CC-MAB performs significantly worse than others and not so much better than random. We concluded possible reasons in conclusion of part 1.

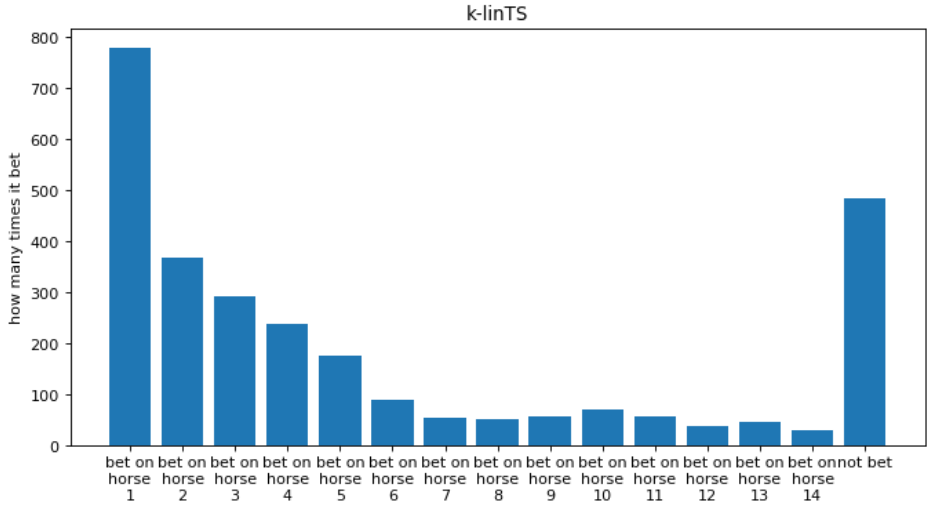

Figure 5.6 How many times each action is played for k-LinTS.

This is the result of one of the runs for k-LinTS. We can see that the frequency of actions played decrease from action 1 to 14. That follows the order we give these actions (ordered by ranking of predicted finishing time). This is exactly the order of how likely each horse win, which is not surprising given how reward function 1 works.

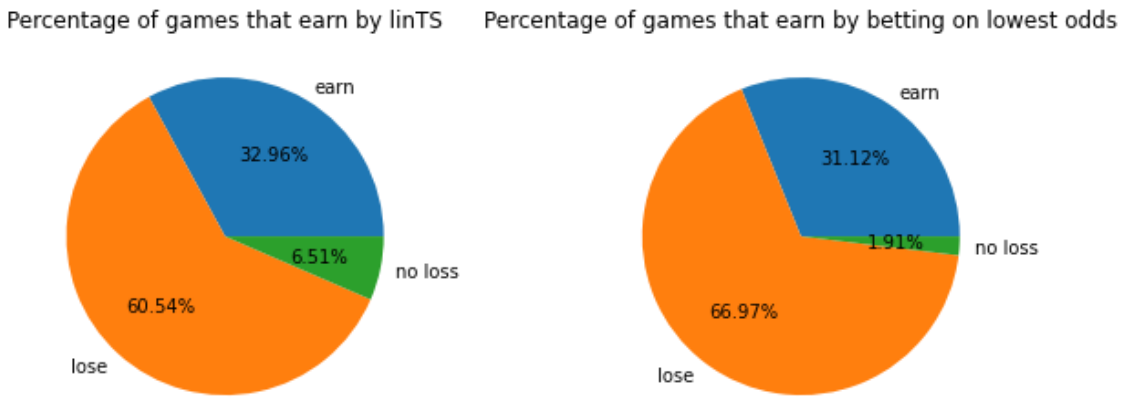

Figure 5.7 percentage of games (2 bet per game) that earn

We can see that the agent is at least on the same level with betting on lowest odds in terms of earnings. And the agent can choose not to bet sometimes, which gives it advantage and there are more races with no loss of money.

- **Reward function 2**

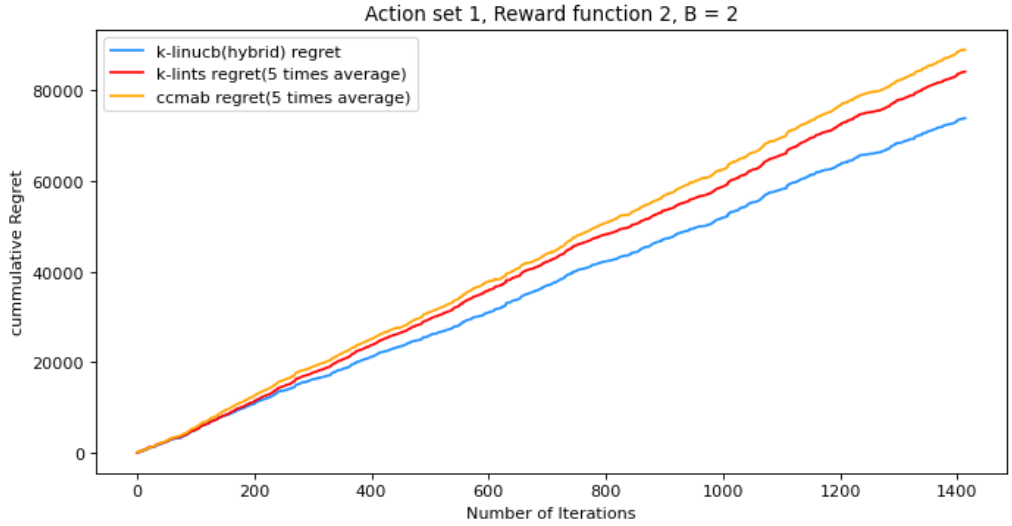

Figure 5.8 Cumulative regret for action set 1, reward function 2

This time LinUCB performs better than LinTS. It could be that using actual cash change as reward, the range of rewards can be large (from -10 to more than 10). While for LinUCB, it works on mean reward that it's not so affected by this issue.

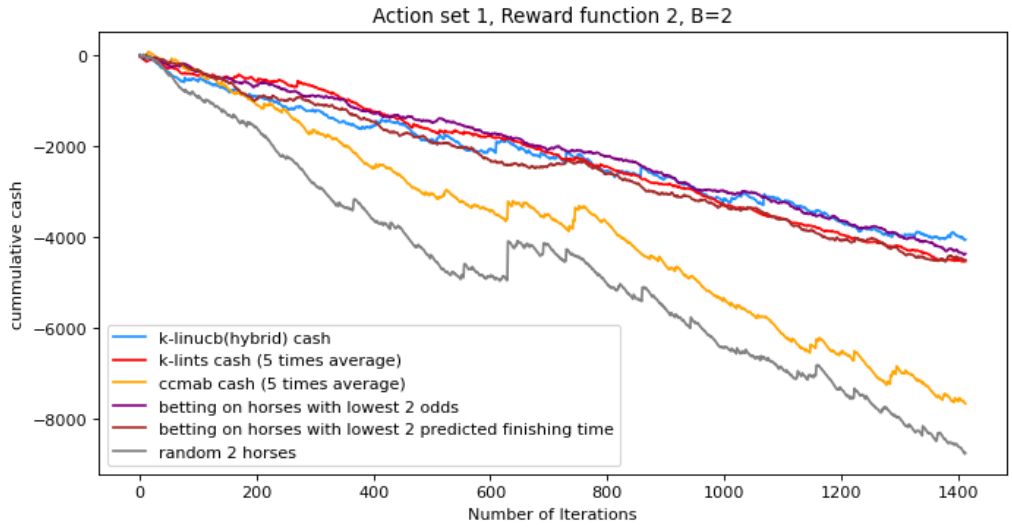

Figure 5.9 Cumulative cash balance change for action set 1, reward function 2

Still there is no algorithm that can earn money at the end. The best performing algorithm doesn't outperform general intelligence and our prediction by much. We can see that it does earn much very often but still fail to maintain cash balance in the end.

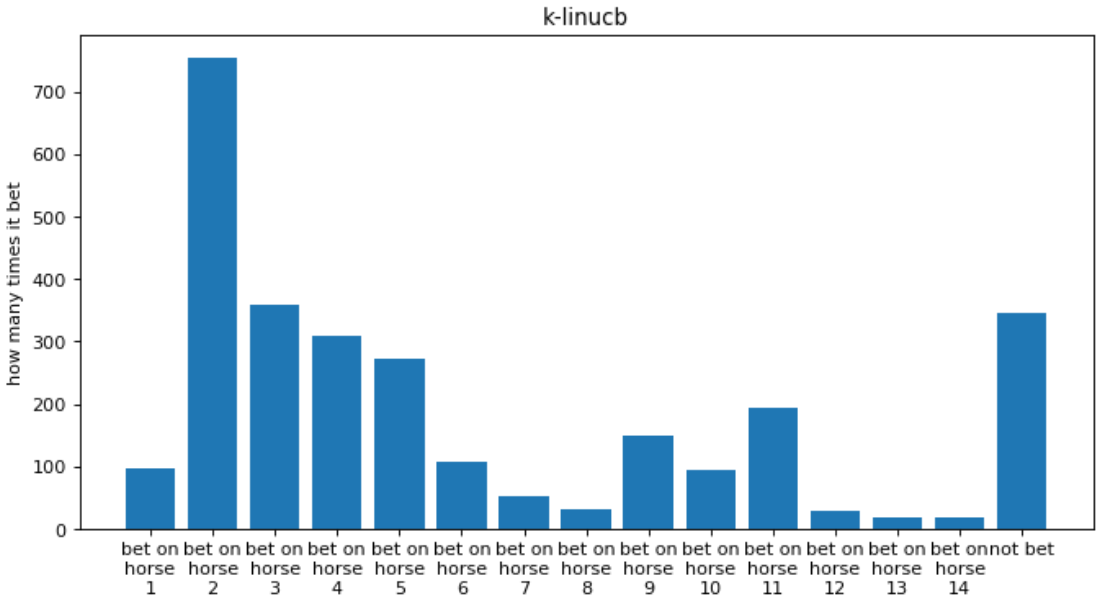

Figure 5.10 How many times each action is played for k-LinUCB.

Most played actions are 2-5. And surprisingly action 1 played less than not bet, this might be related to the reward function we give not betting or the order of our dataset.

Action 2 is significantly played more frequent than others. This is sensible as action 2 will highest average return within top 3 (around -1.23, for action 0 and 2: -1.79, -1.86) while maintaining high probability in top 3 (47%). This shows that if we use actual gain as reward, the algorithm might be able to take those with higher estimated cash. However, this certainly would not be enough as for action 2, the average returns are still negative.

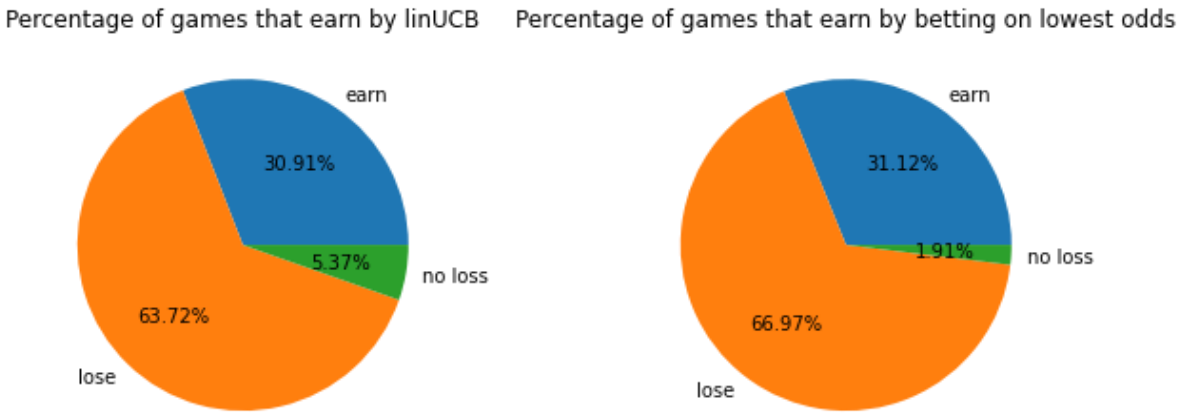

Figure 5.11 percentage of games (2 bets per game) that earn

Similar to results of previous part, the times the agent earns is close to that of betting on lowest odds and there are less races that lose money.

- **Reward function 3**

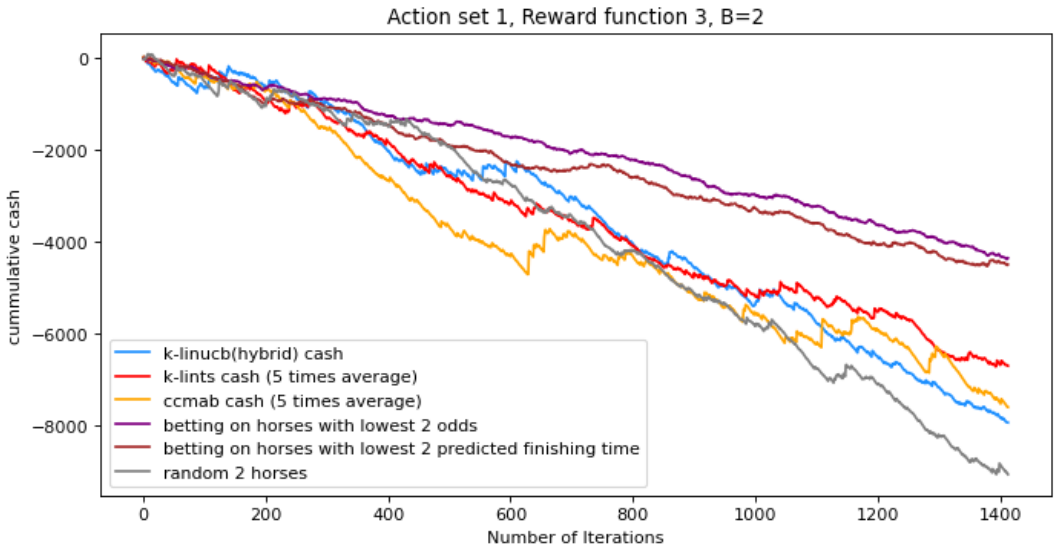

Figure 5.12 Cumulative cash balance change for action set 1, reward function 3

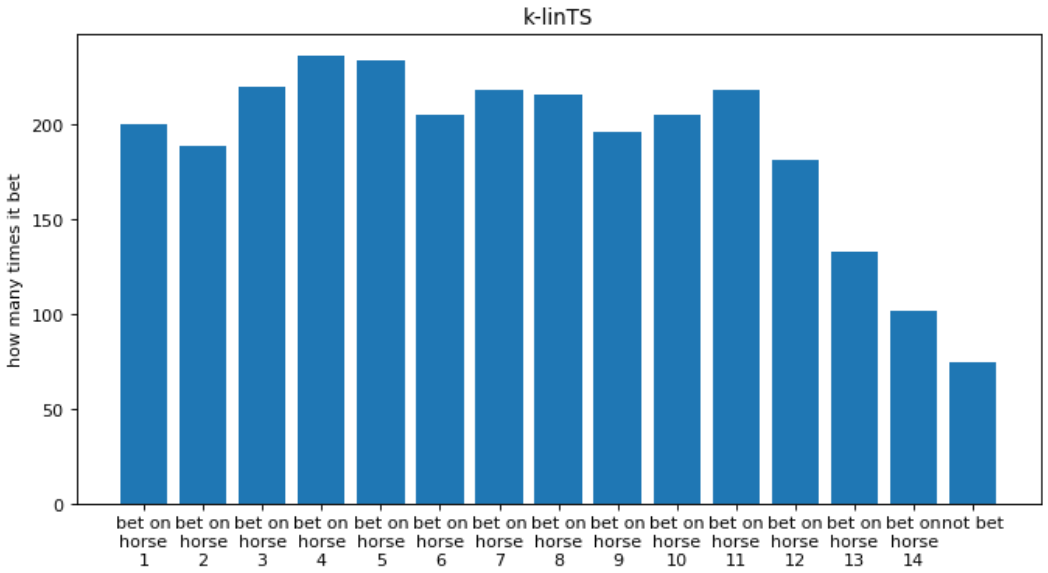

Figure 5.13 How many times each action is played for k-LinUCB.

For this time, the result is not good. The reason is obvious, the actions are equally chosen. The possible reason is stated in 4.2.3.3, that is after division the difference between rewards might be minor, to the point where it doesn't differ much. This is not a suitable approach to address the issue of reward function 2.

Action Set 2

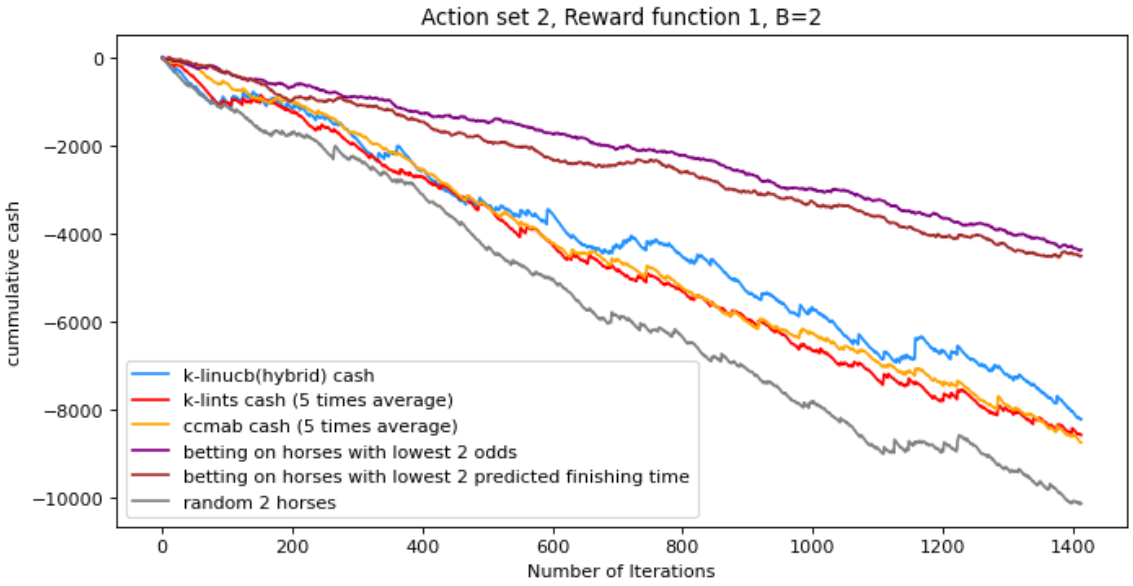

Figure 5.14 Cumulative cash balance change for action set 2, reward function 1

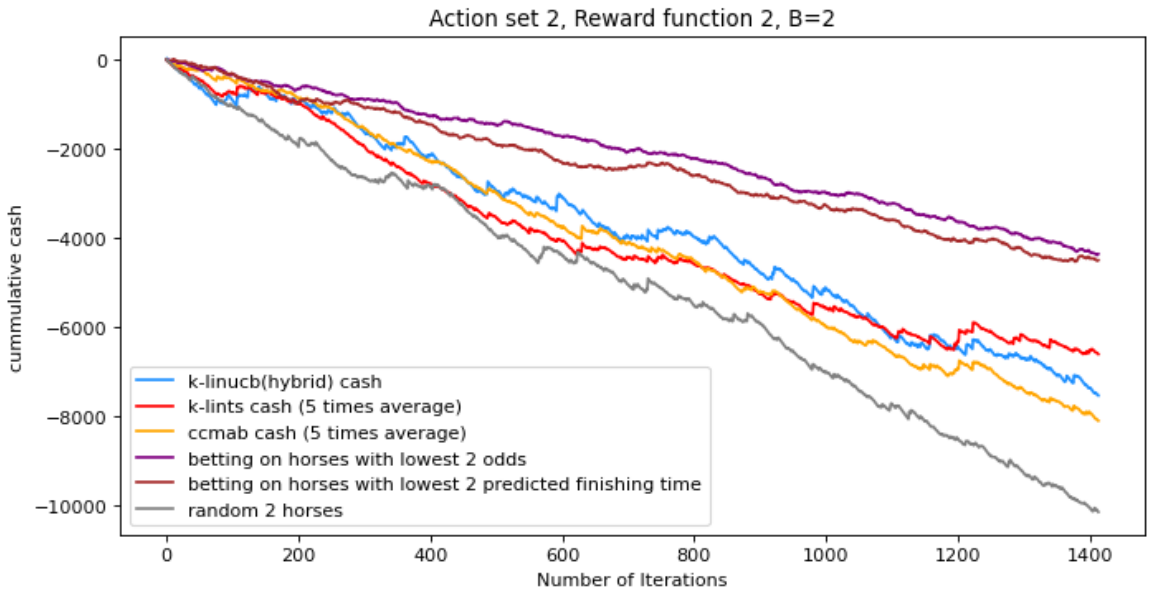

Figure 5.15 Cumulative cash balance change for action set 2, reward function 2

The result is worse than action set 1. They can still perform better than random, which means they are still working. However, it seems that even they all have some parameters not specific to arm but fails to generalize for all horses. This suggests we will at least need some groupings of similar actions like action set 1.

Result and Analysis

From results of reward function 1 and 2, we can see that bandit algorithms work well on playing actions with higher estimated reward. However, this is also the limitation of it. The top actions clearly have highest estimated reward but often they don't give us overall positive return. And bandit algorithms are essentially greedy as they are dependent more on the observed rewards in first few rounds, and for those last ranked horses with higher odds will not be played much as they are seen to have low win rate. Therefore, it will stay with those that win more but have negative average return. However, if we can exploit these options by means like betting more money or don't bet when the odd is low, we can potentially improve the results. And this will be the focus of next part.

We can see from result of action set 2 that we do need to group horses like in action 1 or use better estimator that can generalize well for all different horses.

However, we might also need a better way other than using ranking to group horses, since the estimated reward will greatly depend on the accuracy of the ranking.

We intended to use CC-MAB to test whether grouping horses with close context is a possible approach. However, it doesn't work. In 2.5.1.3, we explained that it works by partitioning context

space into hypercubes. We inspected that the number of hypercubes is 2796, while there are only 2828 actions played in 1414 races. That means not many actions are grouped into same hypercube. This should be due to the dimensions of our context. We tried to reduce the features but features like betting odds take many possible values make the number of hypercubes remain high. This suggests that this simple way of using similarity information doesn't work, especially when we have limited data. We will need better approach than that.

5.1.4.3 Part 2: Attempts to Improve on Direct Betting

In the previous part, we always only bet 10 dollars on each horse. However, as what is mentioned, the betting odds are usually low, and we don't get much by doing so. Therefore, it could be better if we can bet more. And for those with very low odds, we cannot earn much even we win, it would be better if we only bet when the expected earn is above certain threshold. To do that, we take advantage of bandit algorithms.

Overall Flow

There are two improvements we would attempt in this part:

First, since usually when the odds are low, we can't earn much either. Therefore, it would be good if we only bet when the odds are not too low.

Second, in part 1, we just bet 10 dollars on each race, this time we would like to exploit actions those with possibly higher returns.

To do these, we again would use bandit algorithms.

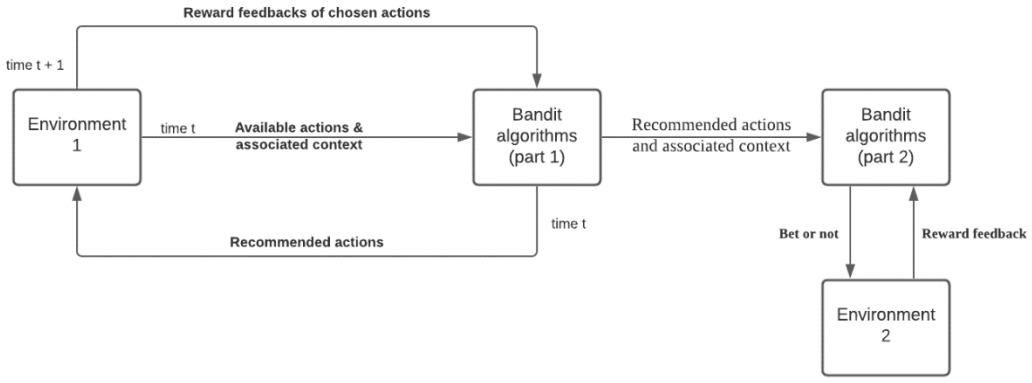

Figure 5.16 Flow Diagram of Part 2

Basically, for actions output from the bandit algorithms in part 1, this time we won't directly take the actions and bet. But rather we would run another bandit algorithms, they would receive the context of the recommended actions and decide whether to take the action. There will be two bandit algorithms running. In particular, we would use LinUCB. Each bandit algorithms are for

different purposes. One is to decide whether to bet $10 which is the basic bet, we would expect it to bet when the money returned is possibly at least $15. Another is to decide whether we should bet twice the basic bet, which is $20, we would expect it to bet when money returned of basic bet is $20.

To achieve these, we would set two different reward functions for each bandit algorithm.

Reward Functions

Reward function for 1st bandit

Since we would want it to bet $10 only when money returned is at least $15, we would penalize losing or return lower than that. We define the rewards for betting as follows:

- $R(bet\ and\ return \geq 15) = return$
- $R(bet\ and\ return < 15) = -10$

For not bet, if the money returned when bet is <15, not bet would be a better choice, we would give it a positive reward of 10. If the money returned when bet is >=15, we would penalize not bet by setting reward to be net gain in money if bet. We define the rewards for not bet as follows:

- $R(not\ bet\ and\ return\ when\ bet < 15) = 10$
- $R(not\ bet\ but\ return\ when\ bet \geq 15) = -\ net\ gain\ if\ bet = return - 10$

Reward function for 2nd bandit

The formulation is similar for betting with $20.

- $R(bet\ and\ return \geq 20) = return \times 2$
- $R(bet\ and\ return < 20) = -20$
- $R(not\ bet\ and\ return < 20) = 10$
- $R(not\ bet\ but\ return \geq 15) = -\ net\ gain\ if\ bet\ \$20 = -(return - 10) \times 2$

*Return refers to the money returned if bet on $10.

Results and Analysis

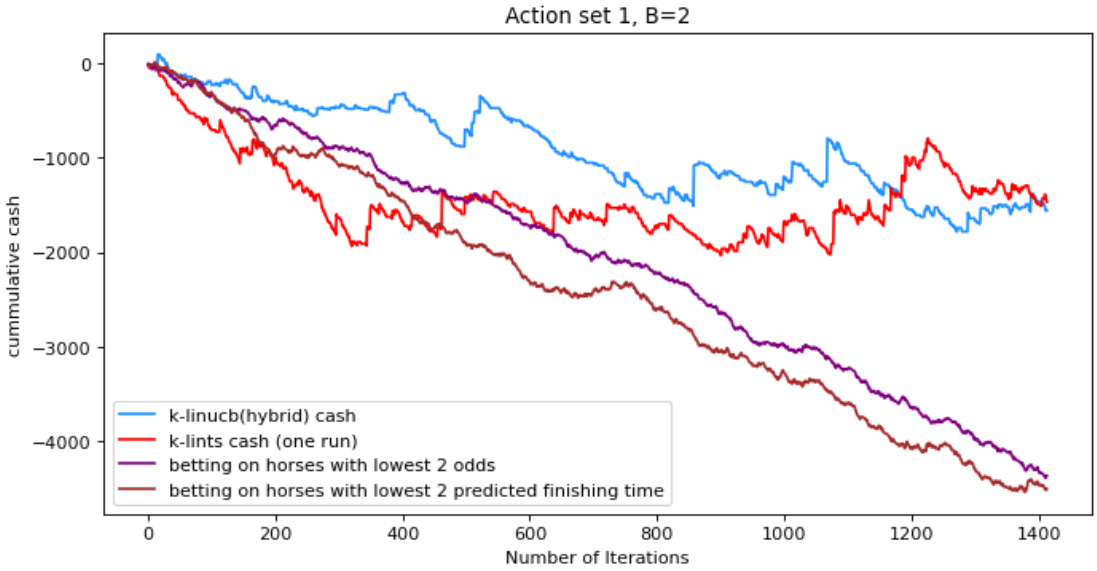

Figure 5.17 Cumulative cash when using actions output of results with reward function 2

Still, we cannot generate profit at the end. When compared with the previous results, we lose less and this time we don't just steadily losing money, we can often have some large gains. However, it cannot maintain the balance and there is always continuous steep descent in the curve. It can hardly to be considered stable.

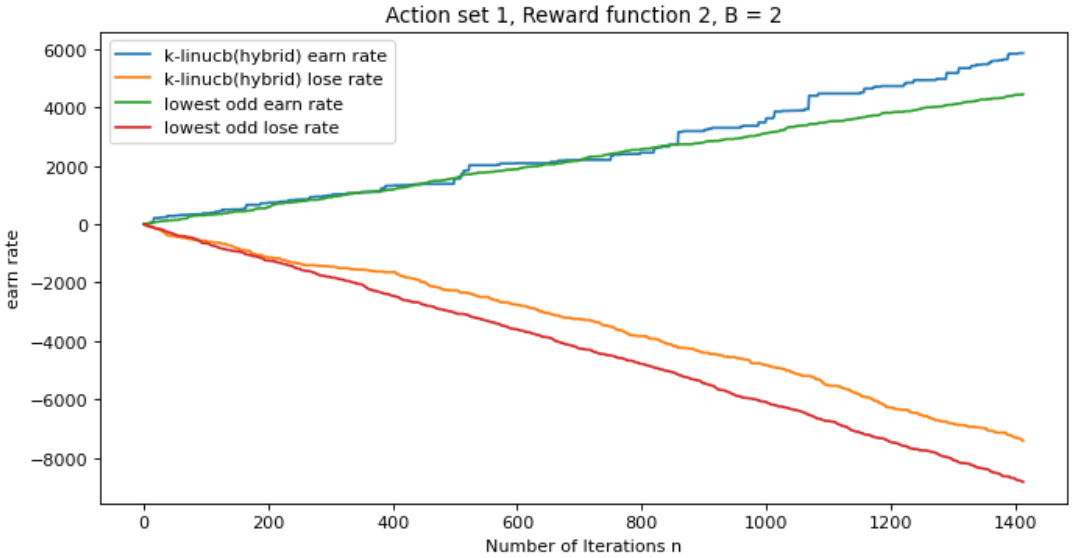

Figure 5.18 Earn rate when using actions output of results with reward function 2

The earn rate or lose rate is better than public intelligence and keeps increasing as time goes on.

5.1.4.4 Part 3: Further Improvements

Use better approximator

In the previous part, we used algorithms with linear models, and they show fine results. In this part, we will see how much we can improve the results by using more complex models especially neural networks. As stated in chapter 2, we would use neural bandit (neural epsilon) and neural ucb.

Regret

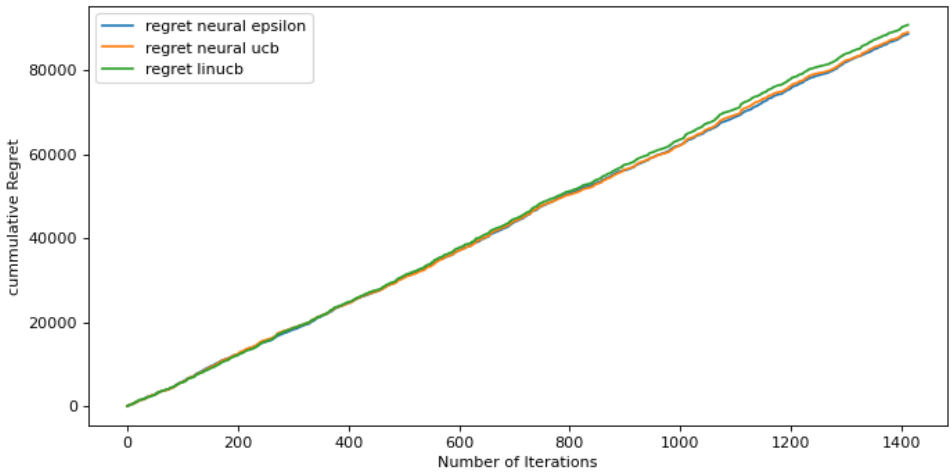

Figure 5.19 Cumulative regret when using new approximator

The regret only slightly improves over linucb

Cash

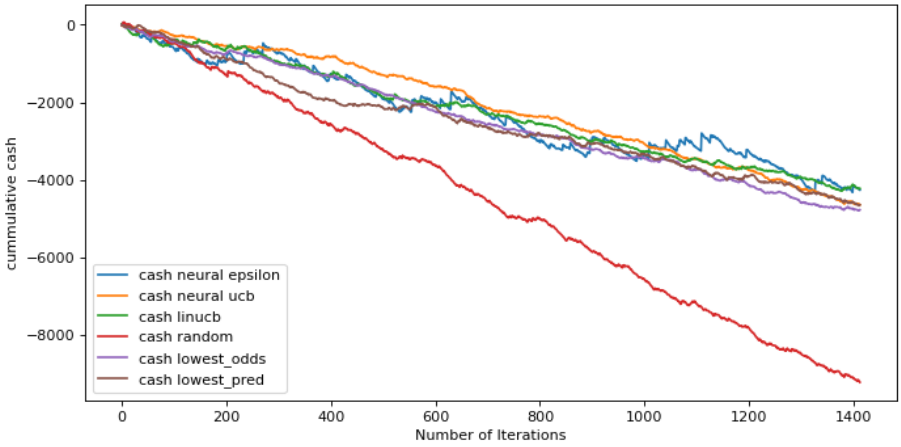

Figure 5.20 Cumulative cash when using new approximator

Both achieve slightly better results than lowest odds and prediction. And perform better than linucb.

Neural bandit

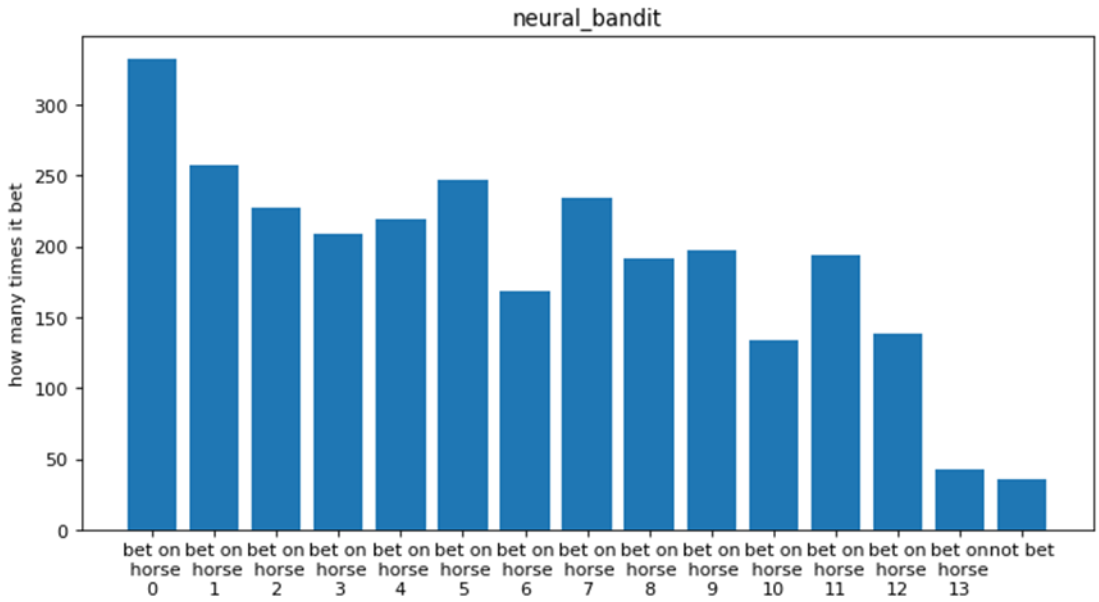

Figure 5.21 How many times each action is played for neural bandit

Since this is using epsilon greedy as exploration mechanism, it has high frequency for each action. However, we cannot see the impact from both the cash balance and proportion of games that earn. A possible reason could be now it is betting more accurately on some high returning options.

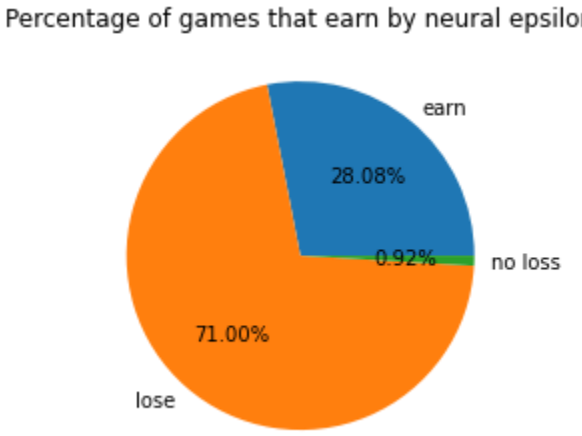

Figure 5.22 Percentage of games that earn by neural epsilon

The proportion of games that earn is similar to previous result of linucb.

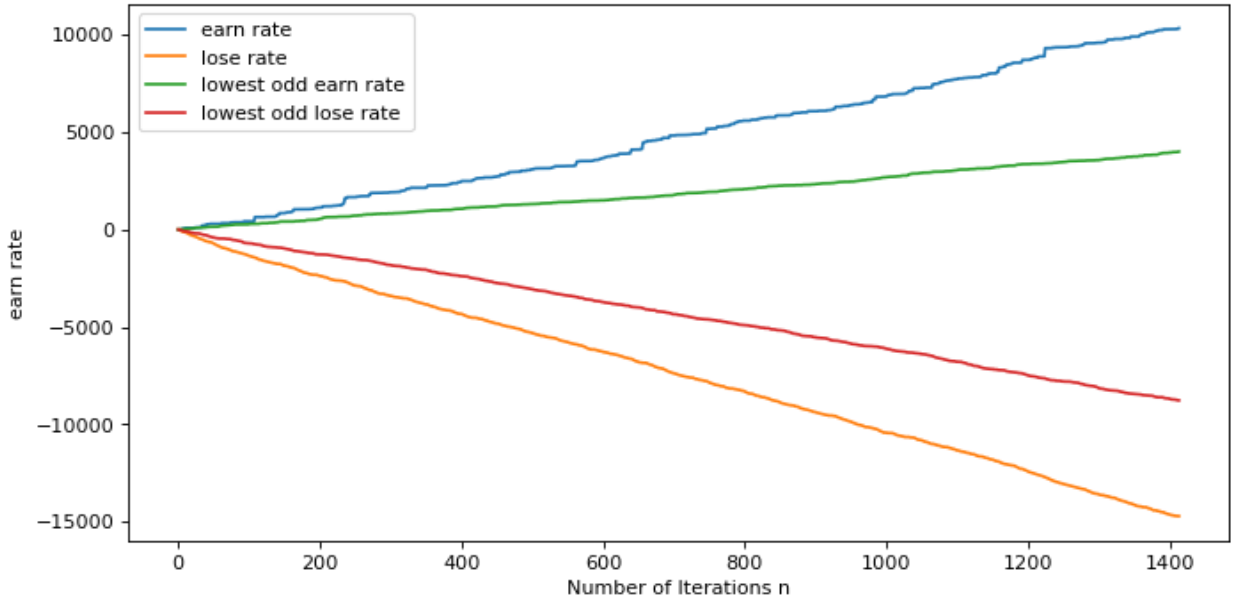

Figure 5.23 Earn rate when using neural epsilon

The earn rate grows over time but the lose rate is not well controlled.

Neural UCB

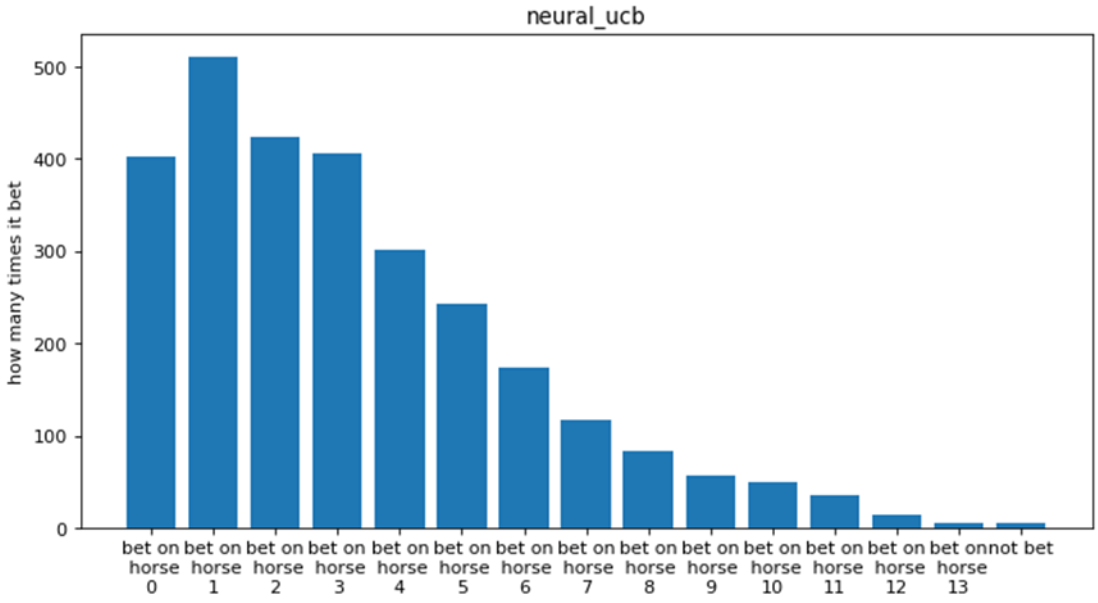

Figure 5.24 How many times each action is played for neural ucb

The frequency of actions chosen follows the trend of win rate. It’s not particularly interesting.

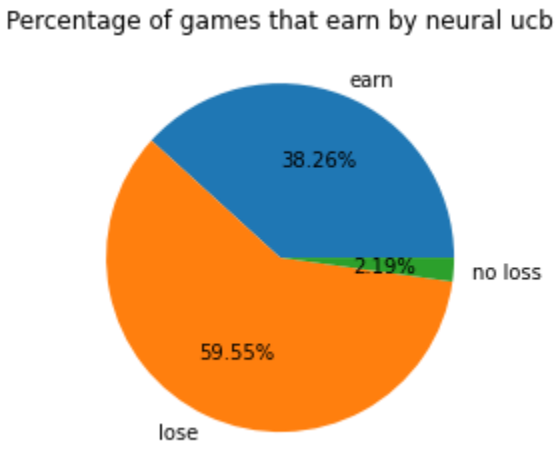

Figure 5.25 Percentage of games that earn by neural ucb

The proportion of games that earn significantly improves over previous result of linucb which is around 30%.

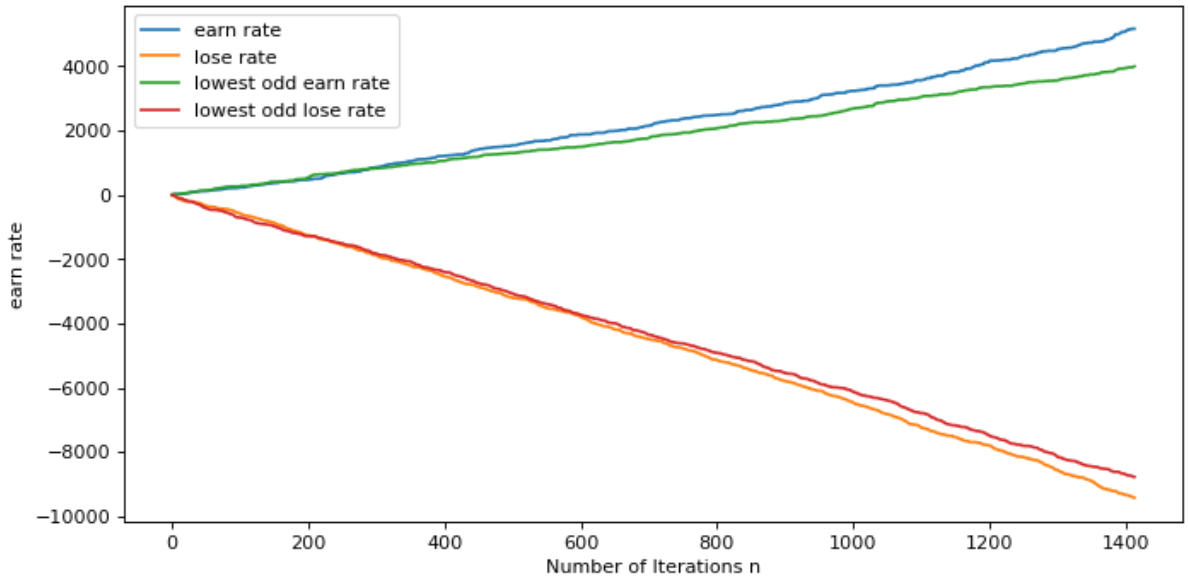

Figure 5.26 Earn rate when using neural ucb

The earn rate also grows over time but the lose rate is not so significant.

Bets on less options

Since from previous result, we can observe that the algorithms bet on top few horses the most. Therefore, we can consider like only top 5 horses.

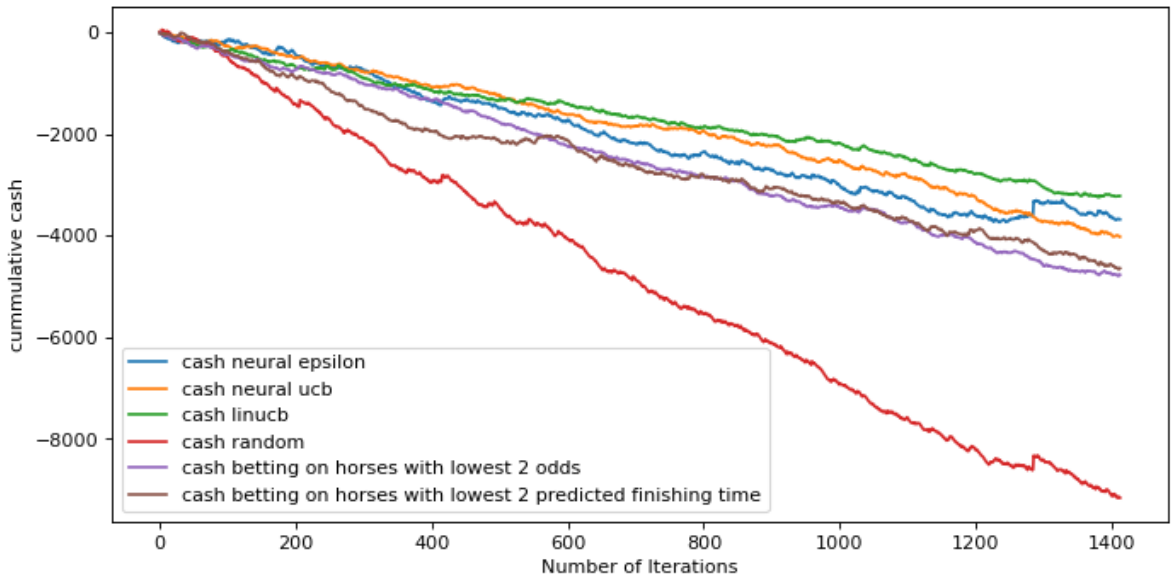

Figure 5.27 Cumulative cash when betting with less options

We can see that there are indeed improvements but not so significant.

Repeating Part 2 with neural ucb

In this part, we do in the same way with part 2 like continue using linucb for two bandits but the base model is swapped to neural ucb.

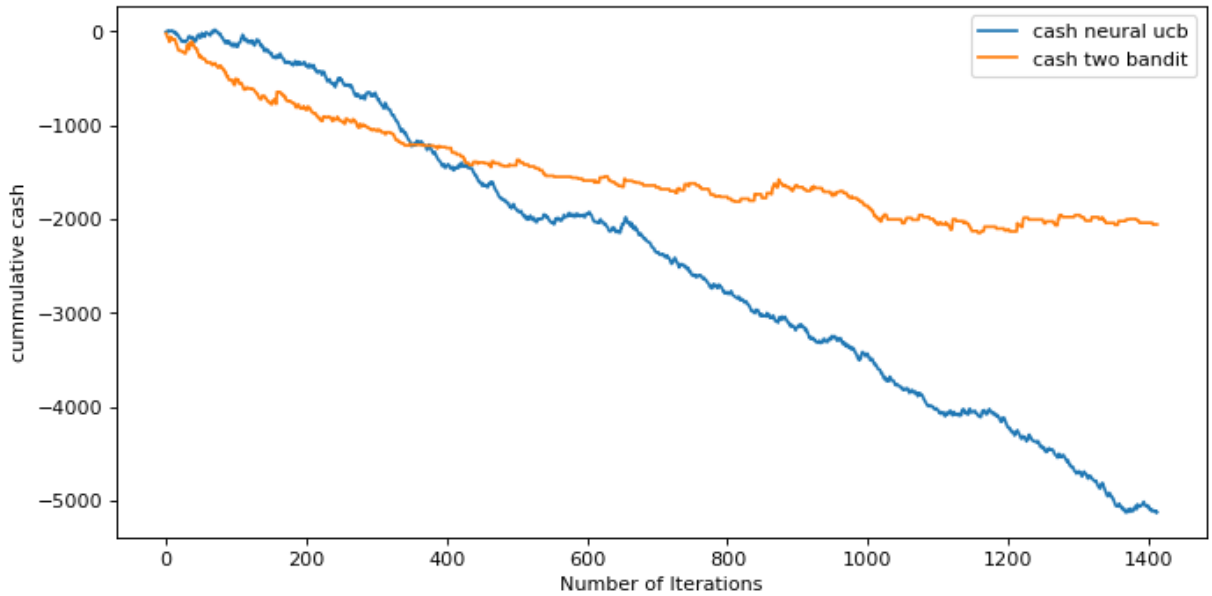

Figure 5.28 Cumulative cash when simulating part 2 using neural ucb

It can maintain the balance. However, we don't see its gaining profit.

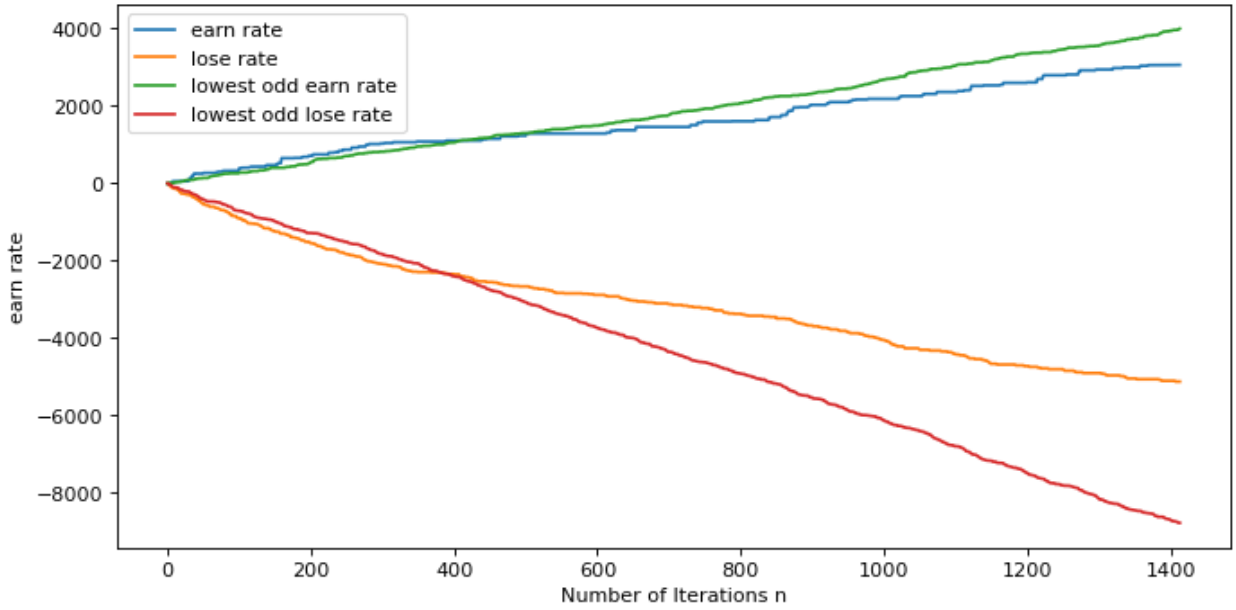

Figure 5.29 Earn rate when simulating part 2 using neural ucb

The earn rate is increasing over time and lose rate decreases over time as well.

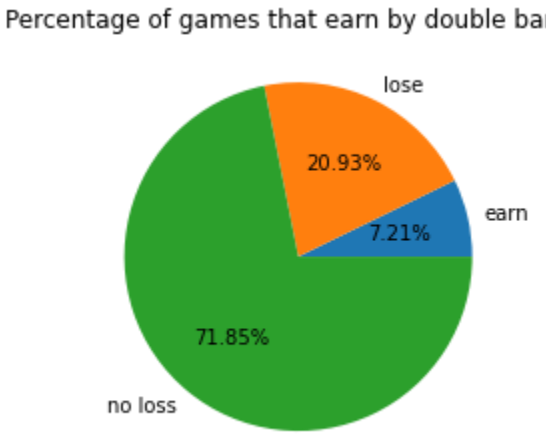

Figure 5.25 Percentage of games that earn by double bandit

Most of the time it doesn't bet and lose is still more than earn.

5.2.5 Summary

It's been hard to define the reward for not betting. In bandit problem, the arms are independent from each other, we could hardly make the reward defined on the outcomes of the other arms. Therefore, in many cases, we just statically set the reward. It's tricky to do so. In some trials, if

we set the reward too high, the agent will tend to choose it. If set not enough high, then it is seldomly picked.

The stochastic bandit algorithms we use are parametric, there are parameters that are related to the property of underlying reward distribution which we don't know much about for horse racing.

It's not so flexible to use bandit algorithm for horse betting. In order to make it bet a different amount of money, we used some tricky constructs. If directly use dollar bets as actions, it won't work as it always fall back to the safest option which is $10 dollar bet. In addition, bandit algorithm is unable to consider state information such as cash balance. We should expect a good strategy should be able to do differently for different situations, e.g., bet bolder when cash balance is high and bet more cautious when cash balance is low. Therefore, this suggests the use of more advanced RL algorithms such as those that consider Markov chain.

5.2 Modeling horse betting as a standard RL problem

5.2.1 Distinction between bandit problem and Markov decision process

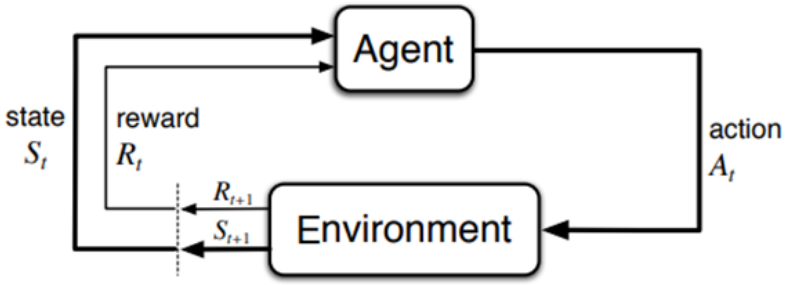

Figure 5.26 Markov Decision Process

A Markov decision process can be defined in 4-tuple:

a. State space S, set of states
b. Action space A, set of chooseable actions at each state
c. Transition probability $P(s'|s,a)$, the probability of choosing action a at state s leading to next state s' (i.e., the next state only depends on current state s and chosen action a, which is called Markov property)
d. $R(s,a,s')$, the immediate reward returned by choosing action a at state s and transitioning to s'

The goal of reinforcement learning for problem defined as Markov decision process is to find optimal policy that maximizes expected total reward over possible trajectories.

On the other hand, bandit problem can be viewed as a Markov Decision Process without state or state transition. It also changes the goal from maximizing expected total reward to maximizing total expected reward as the current state is independent from previous actions and states.

Modelling horse betting as bandit problem is generally fine as there are no dependencies between races and cash balance is irrelevant if we assume we have infinite budget like what we did in previous parts. If we consider the more realistic settings where we have limited budget, then bandit algorithms might not be good enough. In addition to that, we certainly hope the agent to behave differently depending on the amount of cash balance (e.g., bet boldly when we have sufficient money, bet safely when we have barely any money left). Therefore, it might be more interesting to use algorithms that can take state into account and see if how their behaviors differ from bandit algorithms.

5.2.1.1 Environment (Type 1)

In order to compare the performance of different reinforcement learning algorithms easily, the training and testing environments for all algorithms experiments would be constructed in the same way. Besides, we have constructed 2 environments and reward functions for 2 different learning purposes which are learning to bet on 1 horse to win the most with fixed amount of money and with a different amount of money respectively. In this section, the environment with fixed amount of money bet would be discussed.

Betting Rules

Initially a budget of $10000 would be assigned to the agent and each time the agent would select the betting option to bet with $10. Since we are betting PLACE instead of WIN which means if the horse selected by the agent gets into the top 3 places, agent would earn $10 \times$ place odd including the money spent. Otherwise, the agent would lose $10.

Data to Use

We have 1414 races in total, each with 5-14 horses, for those that are not up to 14 horses, we will append with records with horseid as -1 and all other features as 0 to make the dataset to have unified dimensions. In our programs, those with horseid -1 will be ignored. The data would be split into 2 equal halves with the size of 707 and the first half would be used for training and the second part would be used as testing in order to preserve the time order of the horse races held and simulate the horse betting environment as realistic as possible.

Action Space

There will be 15 discrete actions in total where action 0 to action 14 represents the horses in each race and action 15 is the option that not betting. For some races, there may not be 14 horses in total

and thus, dummy data would be filled up for those races to ensure there are total of 14 options for agent to select. Horses in each race are ordered by the finishing time prediction by the random forest model in chapter 4 in ascending order.

Observation Space

The observation space includes the features of the horses in a race and the cash balance which indicates the available amount of money that the agent can spend. The input features are listed as follow:

- Last moment place odds (It is close to the final odds which can roughly tell how much we will earn if we bet right)
- Last 10 minutes exponential moving average (we can tell whether the odds are decreasing/increasing which reflects confidence in this horse),
- Rankings
    - Initial odds provided by HKJC, which reflects their predictions
    - Last moment odds before betting ends, this reflects confidence of gamblers, we call it public intelligence
    - Predicted finishing time from our random forest model
    - Since bandit algorithms consider each action to be independent and don't consider the correlation between actions (which is not true in our case as the horses are competing with each other), hopefully adding information about ranking can help addressing the problem.
- The ratio between finishing time of each horse with that of the horse predicted to ranked 1 place ahead of it in ranking of predicted finishing time, which is to capture the relative difference between each place.
- The frequency of getting 1^{st} placed, 2^{nd} placed, and 3^{rd} placed of the horse which can be referred as a past performance of the horse and used to determine the overall performance and the probability of getting into the top 3 places in the next race.
- The prediction error of that record computed based on predictions of different decision trees in the random forest including the upper bound, lower bound and the range of the errors. As mentioned in section 4.7, betting on those horses that satisfy the error checking only is a good way to avoid betting on the horse which has lower confidence in it.

For the dummy data that filled into each race would have values of -1 for all features.

Steps

During each step, the agent would first collect the state from the environment which are the features of the 14 horses in the next race and the cash balance of the agent. Then the agent would select the action from the action space with the computation of its algorithm and place a fixed amount of money ($10) to bet. The environment would return the corresponding reward to the agent and the terminating state of the game would be determined.

Terminating State

In general, a betting game ends when there is no more horse race, or the cash balance is used up. In order to boost the performance and the ability to control the budget of the agent, a terminate state which is stopping the betting games when the cash balance is less than 9000.

5.2.1.2 Environment (Type 2)

Last semester we have tried to train the agent to learn a good betting strategy based on the random forest result while due to the fact that the accuracy of finishing time prediction is low and the amount of data is not enough, the performance of the betting strategy is not good. In this semester, we will try to let the agent to learn bet more money on the options with higher confidence in order to maximize the profits. In this section, the environment with dynamic amount of money bet would be discussed.

Betting Rules

Initially a budget of $10000 would be assigned to the agent and each time the agent would select the betting option to bet with a certain amount of money. Since we are betting PLACE instead of WIN which means if the horse selected by the agent gets into the top 3 places, agent would earn the money used to bet × place odd including the money spent. Otherwise, agent would lose all money used to bet.

Data to Use

The data used in this environment is same as the data that used in type 1 environment. The data would be split into 2 equal halves with the size of 707 and the first half would be used for training and the second part would be used as testing in order to preserve the time order of the horse races held and simulate the horse betting environment as realistic as possible.

Action Space

There will be 75 discrete actions in total which are constructed by the amount of money bet and the horse options. For instance, 5 lowest amount of money is provided which are $10, $20, $30, $40 and $50 since the HKJC only provides money bet options which is multiply of 10. The agent can choose betting on 1 out of 14 horses and select the amount of money to bet on it which has a total of 70 discrete actions. Besides, a not betting option is provided and in order to ensure the probability of selecting this option and betting on horses are the same, 5 more actions would be added, and all of these are resulting in not betting and thus, there would be a total of 75 options included in the action space.

Observation Space

The observation space in this environment is same as the observation space used in type 1 environment. The observation space includes the features of the horses in a race and the cash

balance which indicates the available amount of money that the agent can spend. The input features are listed as follow:

- Last moment place odds (It is close to the final odds which can roughly tell how much we will earn if we bet right)
- Last 10 minutes exponential moving average (we can tell whether the odds are decreasing/increasing which reflects confidence in this horse),
- Rankings
- Initial odds provided by HKJC, which reflects their predictions
- Last moment odds before betting ends, this reflects confidence of gamblers, we call it public intelligence
- Predicted finishing time from our random forest model

Since bandit algorithms consider each action to be independent and don't consider the correlation between actions (which is not true in our case as the horses are competing with each other), hopefully adding information about ranking can help addressing the problem.

- The ratio between finishing time of each horse with that of the horse predicted to ranked 1 place ahead of it in ranking of predicted finishing time, which is to capture the relative difference between each place.
- The frequency of getting 1st placed, 2nd placed, and 3rd placed of the horse which can be referred as a past performance of the horse and used to determine the overall performance and the probability of getting into the top 3 places in the next race.
- The prediction error of that record computed based on predictions of different decision trees in the random forest including the upper bound, lower bound and the range of the errors. As mentioned in section 4.7, betting on those horses that satisfy the error checking only is a good way to avoid betting on the horse which has lower confidence in it.

For the dummy data that filled into each race would have values of -1 for all features.

Steps

During each step, the agent would first collect the state from the environment which are the features of the 14 horses in the next race and the cash balance of the agent. Then the agent would select the action from the action space with the computation of its algorithm and place a fixed amount of money ($10) to bet. The environment would return the corresponding reward to the agent and the terminating state of the game would be determined.

Terminating State

In general, a betting game ends when there is no more horse race, or the cash balance are used up. In order to boost the performance and the ability to controlling the budget of the agent, a terminate state which is stopping the betting games would be defined. However, since the amount of money bet would be larger than the amount of money used in type 1 environment, the potential loss would

also greater than that, thus $8000 would be set as the lowest limit of cash balance to stop the horse betting game.

5.2.1.2 Reward Functions

Based on the experience in defining reward function for the multi-armed bandit algorithm, the rewards for the agent would be computed based on the dollar lost or gain by the agent in order to satisfy our purpose that maximizing the profitability of the betting strategy

On the other hand, this semester we have computed the confidence level of each finishing time predictions, and the results shown in section 4.6 have proved that the error checking of the predictions would be a good reference for deciding to bet or not. Thus, we would the error checking features would also be considered in the reward function, and it is defined as follows.

- $R(Bet\ any\ of\ top\ 3\ horses\ correctly) = dollar\ bet \times betting\ odd\ of\ betted\ horse$
- $R(Bet\ any\ of\ top\ 3\ horses\ correctly\ and\ error\ range\ <\ mean) = (dollar\ bet \times betting\ odd\ of\ betted\ horse) \times ((dollar\ bet\ /\ 10)\ +\ 0.5)$
- $R(Bet\ wrong) = -dollar\ bet$
- $R(Bet\ wrong\ and\ error\ range\ >\ mean) = -dollar\ bet \times ((dollar\ bet\ /\ 10)\ +\ 0.5)$

Dollar bet is chosen to be $10 as it is the minimum amount of money to bet.

For reward for not bet, we set it to be slightly higher than the average amount of money change when betting randomly, which is -3.2.

- $R(Not\ bet) = -3$

As the actual cash return is used and it may be exaggerated when the betting options are considered as not good, the variance of the rewards is potentially high, which hinders the performance of the agent.

5.2.1.3 General Procedures

The Procedure of the experiment consists of 2 parts. In the first part, the environment with the fixed amount of money which expects the agent can focus on the action selection without any consideration of the amount of money bet. In the second part, the environment which allows the agents to choose different amounts of money bet on each race and expects it can make improvements over the first environment.

Metrics

In order to evaluate the performance of the reinforcement learning algorithms, the cumulative rewards in both training phrases would be the metrics. Normally the cumulative would be expected to converge to a value which refers to the success of the training part. The win rate of the agent in both training and testing phrase would be displayed in order to evaluate the accuracy of betting on the correct horse in each race which is also one of our objectives of the experiment.

5.2.1.4 Challenge of Horse Betting

One of the challenges is mentioned in section 5.4.1.4 which the PLACE odd of horses in each race is very low since the win percentage of betting PLACE is much greater than betting WIN. In contrast, this may greatly affect the profit gain because if the agent loses in 1 race, it needs to bet on the correct horse for average 3 to 4 races to recover the loss.

Another challenge is that since dataset would be split into training and testing data which are formed by different races with different horses, the agent performs well in training phrase does not means that it would also perform very well in the testing phrase. Thus, an optimal strategy for both training and testing data needs to be found but it would increase the level of difficulty of the experiment.

5.2.2 Deep Q Network

5.2.2.1 Hyperparameters

Discount Factor

The discount factor in reinforcement learning is used to discount the future rewards which reflects that the earlier action would have a less and less impact on the future action and rewards. As mentioned in section 2.5.2.1, the discount factor γ would usually be set between 0 and 1 where the closer of γ to 1, the closer relation between early reward and future rewards is. In the previous project, a γ of 1 is used to train the agent in horse betting since number of races T is finite and the importance of both future bets and early bets are identical [16]. In this project, we would also use 1 as a discount factor and thus, the agent could treat all races at the same level and learn how to win all the races.

Experience Replay Buffer

As mentioned in section 2.5.2.2, the Experience Replay is a technique used in deep Q Network to enhance the stability of the updating of the neural network by storing experience in the replay buffer. If the size of the buffer increases, the performance would be improved [22]. While the buffer size is too large, the process would be slowed down and require a large size of memory. In order to strike a balance, a size of 100000 would be set as the buffer size of our experiment.

Structure of Q Network

The purpose of using Neural Network in Deep Q Learning is to solve the problem of lacking generalization in Q Leaning. The Q Network that is used in this experiment has 2 hidden layers with 64 nodes and uses the ReLu activation function in each layer There will be 309 nodes at the input layer for both environments, and 15 output nodes and 75 output nodes for type 1 and type 2 environment respectively.

5.2.2.2 Results and Analysis (Type 1)

In this section, the result of horse betting trained Deep Q Network with the type 1 environment (fixed amount of money bet) would be discussed. Overall, the results of the testing phrase are acceptable.

Cumulative Reward

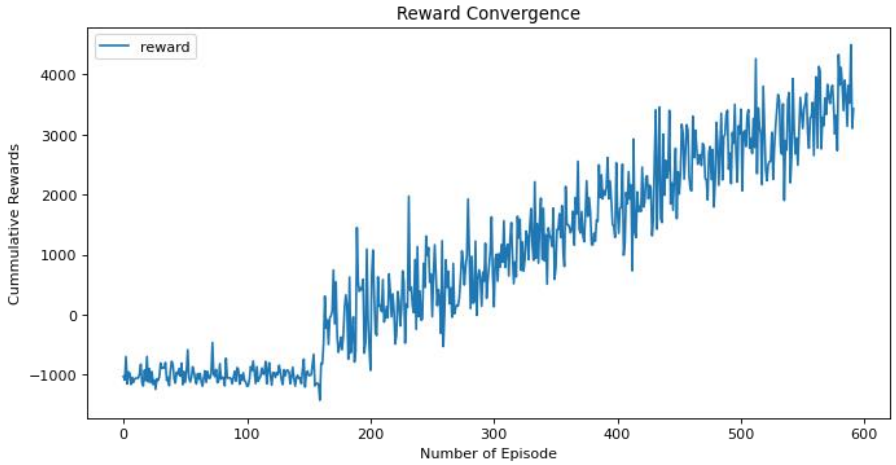

Figure 5.27 Reward Convergence of DQN with Environment 1

Figure 5.27 shows the cumulative reward per each episode during the training phrase which is computed by summing up all rewards in that episode. The reward is increasing while it does not tend to converge after 350724 steps. Since the number of steps is limited in order to avoid serious overfitting.

Cash Balance

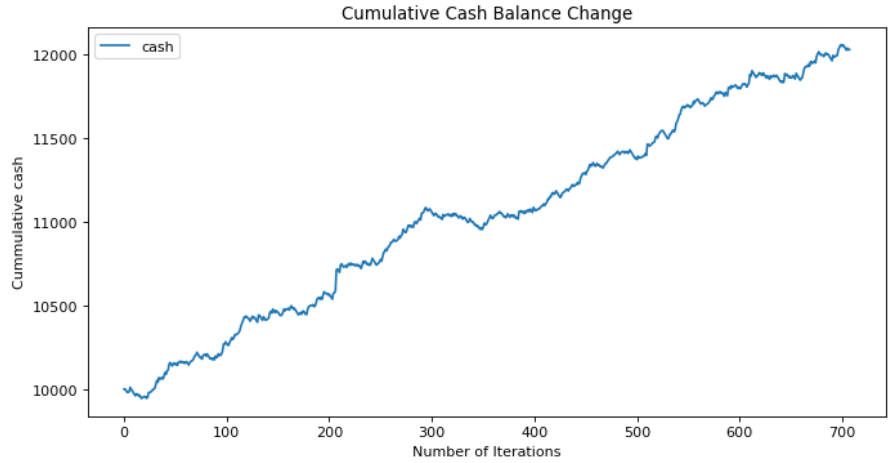

Figure 5.28 Cumulative Cash Balance Change in Training

Figure 5.28 shows the cumulative cash balance that the agent gained with bet with the training data. The agent finally earned $2027 after betting on 707 races and there does not exist huge fluctuation in the chart which shows that the agent learns well with the training data.

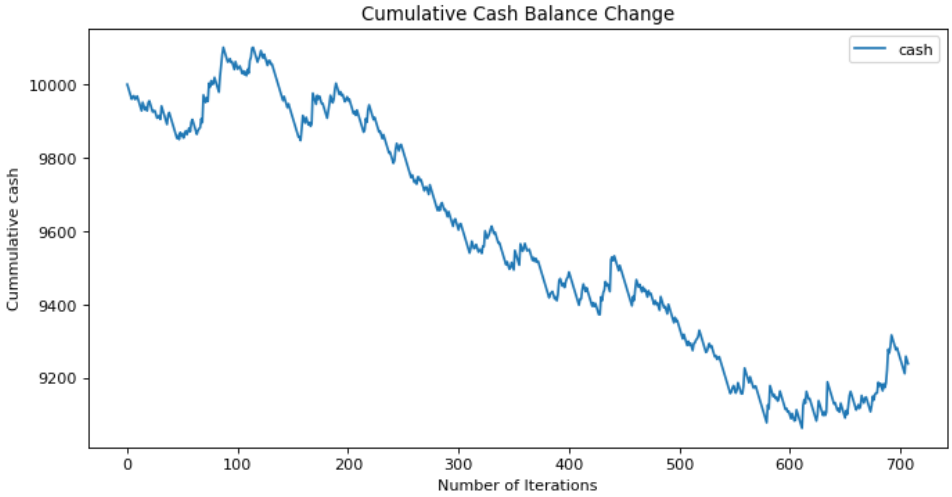

Figure 5.29 Cumulative Cash Balance Change in Testing

In contrast, figure 5.29 shows the cumulative cash balance that the agent gained with bet with the testing data. The agent achieved the highest cash balance $10101 while the cash budget gradually decreases during the testing and finally the agent has $9238 cash remaining. Similar to figure 5.28, the agent does not have a cash balance with huge fluctuation which shows that the agent can still learn how to bet on the correct horse while the accuracy is not good enough.

Frequency of Actions

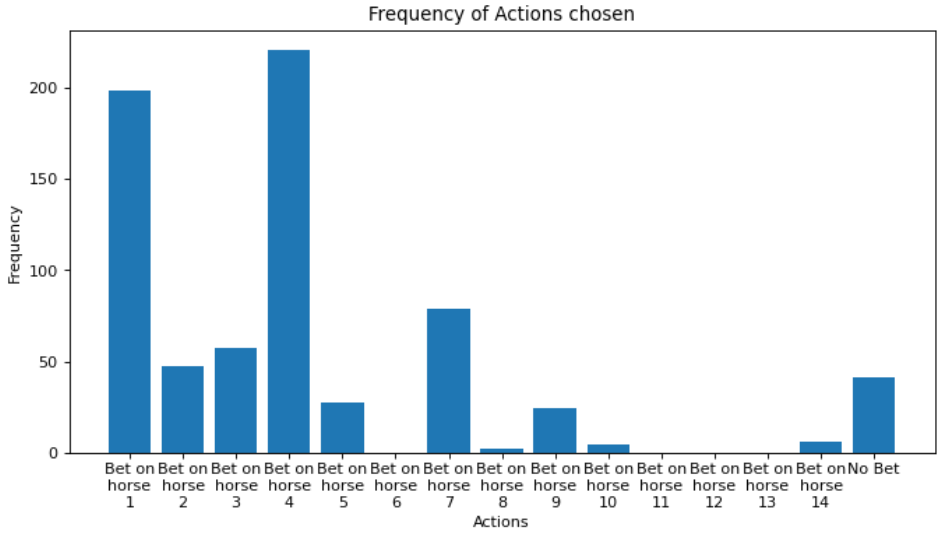

Figure 5.30 Actions Selected in Training

Figure 5.30 shows the frequency of the actions selected by the agent during the training process. Recall that the horses are ordered by their finishing time prediction from the random forest in ascending order which means that horse 1 would be predicted as the fastest horse in each race. From the graph, we can see that the agent tends to choose horses 1 and 4 mostly which is a relatively safe strategy. Also, the frequency of choosing not to bet is relatively low which shows that the profit the agent is not earned simply not placing any bet which satisfied our purpose of setting a negative reward for not betting option.

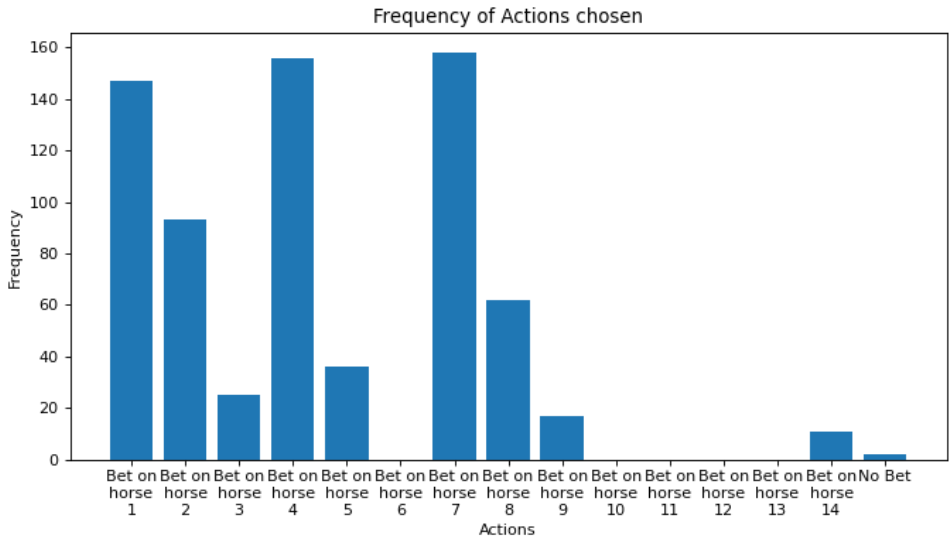

Figure 5.31 Actions Selected in Testing

Figure 5.31 shows the frequency of the actions selected by the agent during the testing process. The main difference between actions selected in training and testing is that horse 7 has been chosen for the most, and the frequency difference between horses 1, 4 and the other options has reduced significantly. Also, the frequency of selecting to not bet has dropped compared with the training process.

Since the agent is more likely to choose betting on the horses that do not have the lowest predicted finishing time rather than simply not betting, which may lead to the result that the cash balance of the agent is much less than the cash balance in training phrase. However, from figure 5.29, some sharp peaks with high positive slopes occur along the curve which can be explained with the reason that the agent selects horses other than horse 7 more frequently and those horses usually have a higher place odd but less likely to win. Thus, the cash balance would fluctuate more usually.

Win Rate

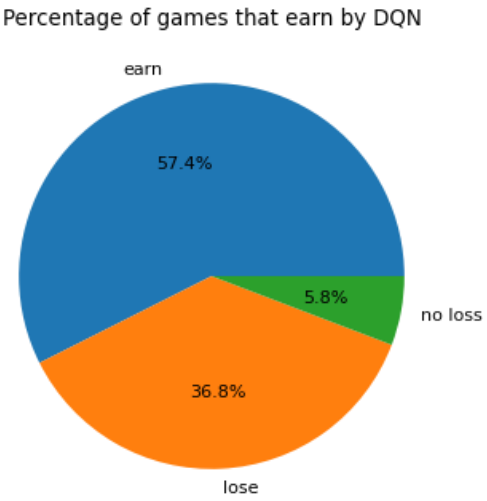

Figure 5.32 Win Rate of Agent in Training

Figure 5.32 shows the percentage of winning, losing and not betting in the training process. The agent achieves a low losing rate of 36.8% and a high winning rate of 57.4% out of 707 races in the training.

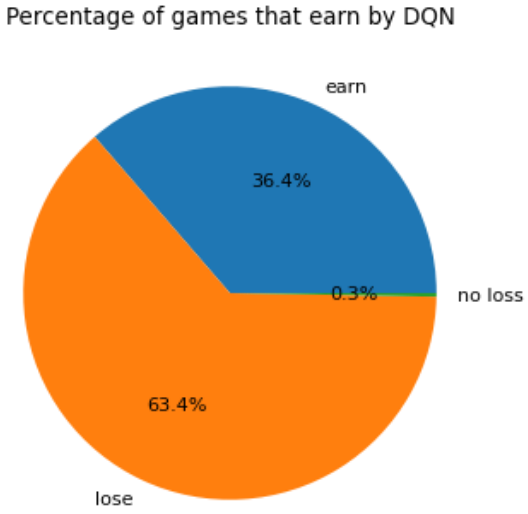

Figure 5.33 Win Rate of Agent in Testing

Figure 5.33 shows the percentage of winning, losing and not betting in the testing process. Compared to figure 5.32, the agent performs worse in testing than in training where it can only achieve an acceptable winning rate of 36.4% out of 707 races in the testing and the percentage of not betting reduces by nearly 95%.

5.2.2.3 Results and Analysis (Type 2)

In this section, the result of horse betting trained Deep Q Network with the type 2 environment (dynamic amount of money bet) would be discussed. Overall, the results of the testing phrase are acceptable while the agent performs worse than the previous model.

Cumulative Reward

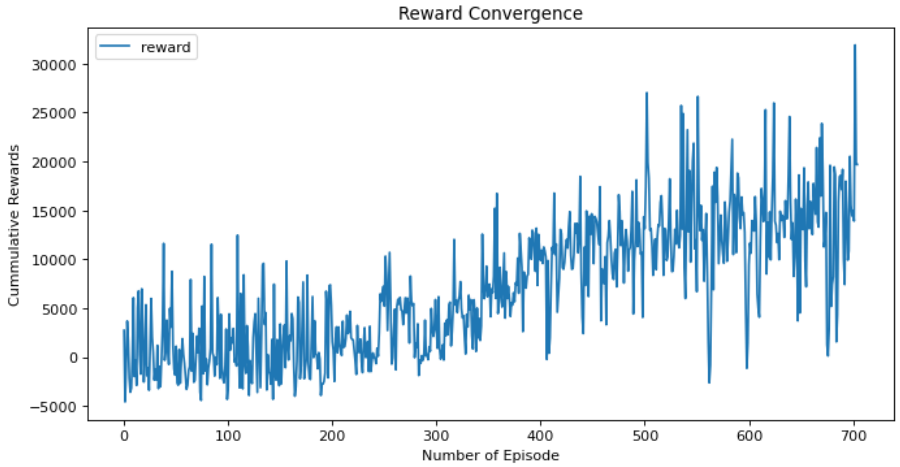

Figure 5.34 Reward Convergence of DQN with Environment 2

Figure 5.34 shows the cumulative reward convergence of DQN which is increasing and converging after 350724 steps while the variance of the rewards is very large. One reason that may lead to the large variance is the reward function could produce rewards with a large variance since the purpose of the type 2 environment is to train the agent to bet on different amounts of money. As well as the dollar bet is one of the components affecting the rewards, thus, the cumulative would have a high variance.

Cash Balance

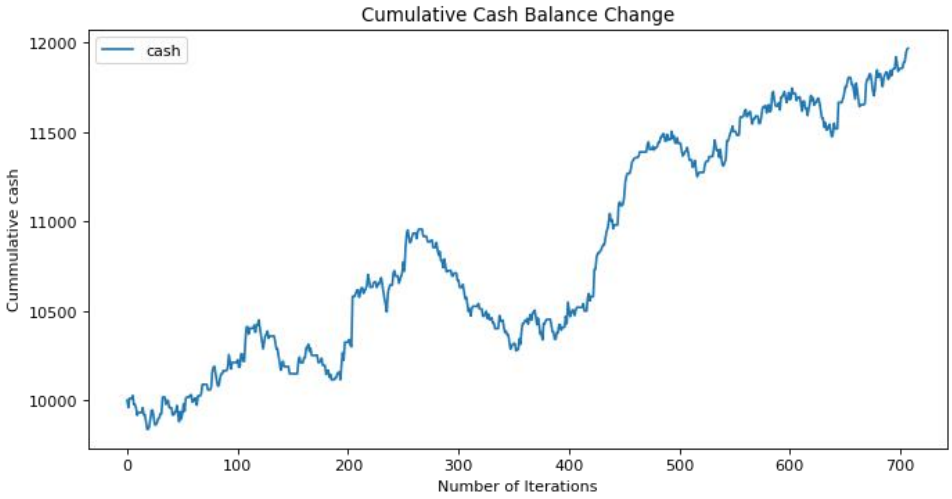

Figure 5.35 Cumulative Cash Balance Change in Training

Figure 5.35 shows the cumulative cash balance that the agent gained with bet with the training data. The agent finally earned $1968 after betting on 707 races which is slightly less than the one with environment 1. Besides, the slope is not as flat as the previous one, which shows that the agent does bet with more money while sometimes it earns or loses a lot at once and this achieves our expectations.

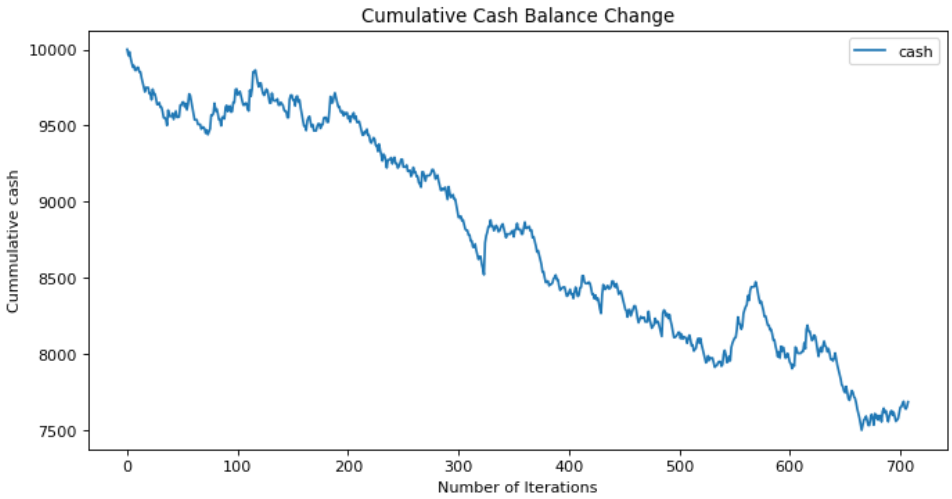

Figure 5.36 Cumulative Cash Balance Change in Testing

Similar to the previous performance in section 5.2.2.2, figure 5.36 shows the cumulative cash balance that the agent gained with bet with the testing data where the agent loses $2318 at the end.

The agent performs much worse than the previous model since most of the time the amount of loss is amplified because of the low accuracy of the finishing time prediction which is shown by a cash balance with huge fluctuation from figure 5.36.

Frequency of Actions

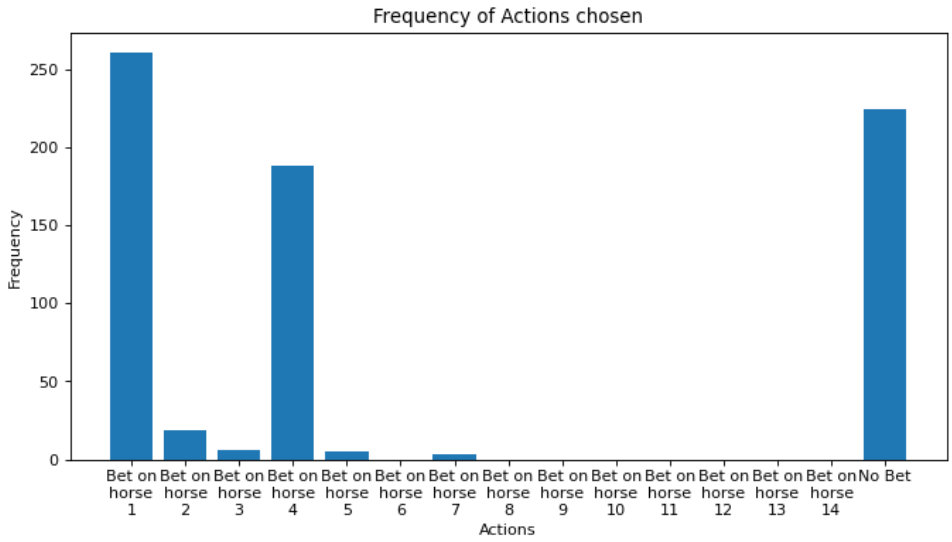

Figure 5.37 Actions Selected in Training

Figure 5.37 shows the frequency of the actions selected by the agent during the training process. Recall that the horses are ordered by their finishing time prediction from the random forest in ascending order which means that horse 1 would be predicted as the fastest horse in each race. Similar to the previous, we can see that the agent tends to choose horses 1 and 4 mostly which is a relatively safe strategy while the difference is that the agent selects to not bet more frequently. This shows that the agent is trained to bet more safely than the model.

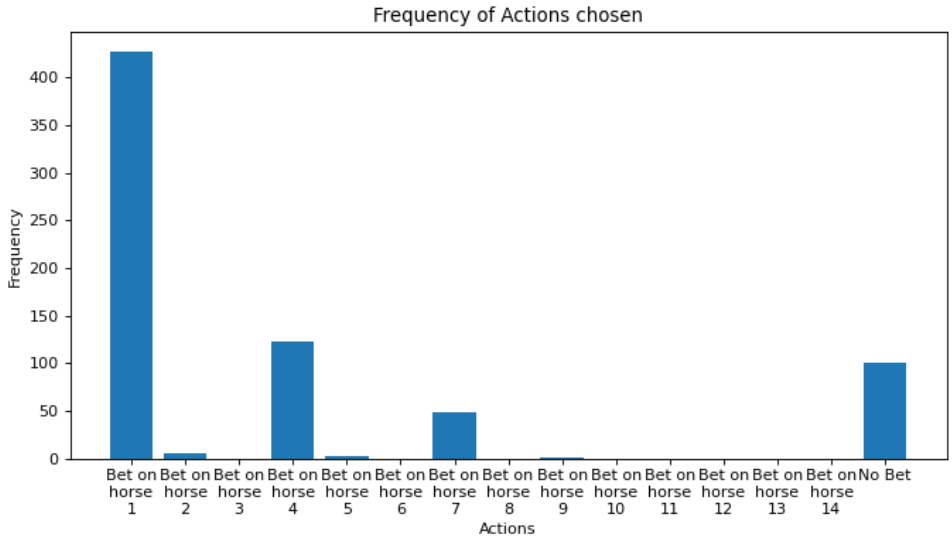

Figure 5.38 Actions Selected in Testing

Figure 5.38 shows the frequency of the actions selected by the agent during the testing process. The main difference between actions selected in training and testing is that horse 1 has been chosen for the most, and the frequency of choosing all other actions has reduced significantly.

This time the agent tends to bet on the horse with the lowest predicted finishing time, which is the safest betting strategy, while it would still lead to a negative profit as mentioned in chapter 4.

Amount of Money Bet

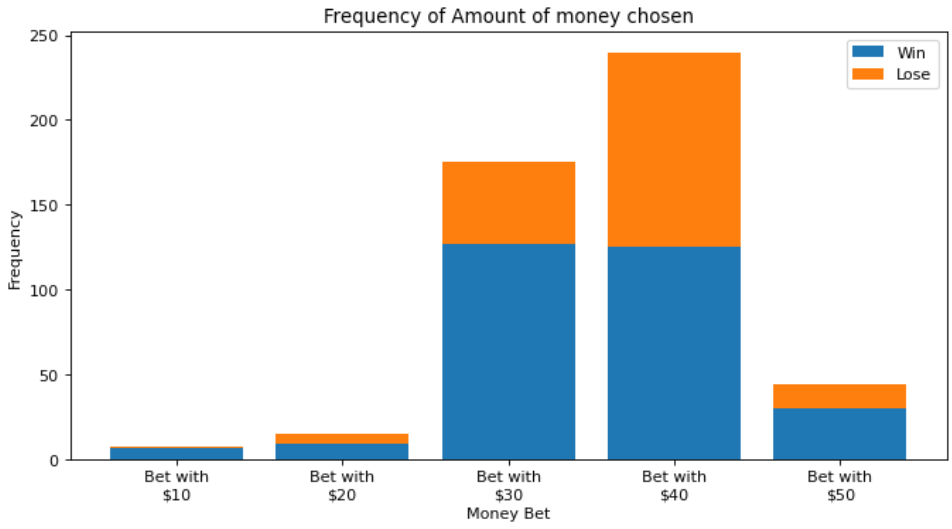

Figure 5.39 Frequency of Amount of Money Bet in Training

Figure 5.39 shows the frequency of different amounts of money bet and the win ratio of each amount of money in the training process. From the graph we can see that the agent bet with $30 and $40 the most while the win ratio is around 60.722%. However, our expectation is that the agent will be able to bet more money when the horses have a higher probability to win and vice versa. Thus, we expect that the win ratio of betting with $10 would be lower than the others while the result shows the agent cannot learn as expected.

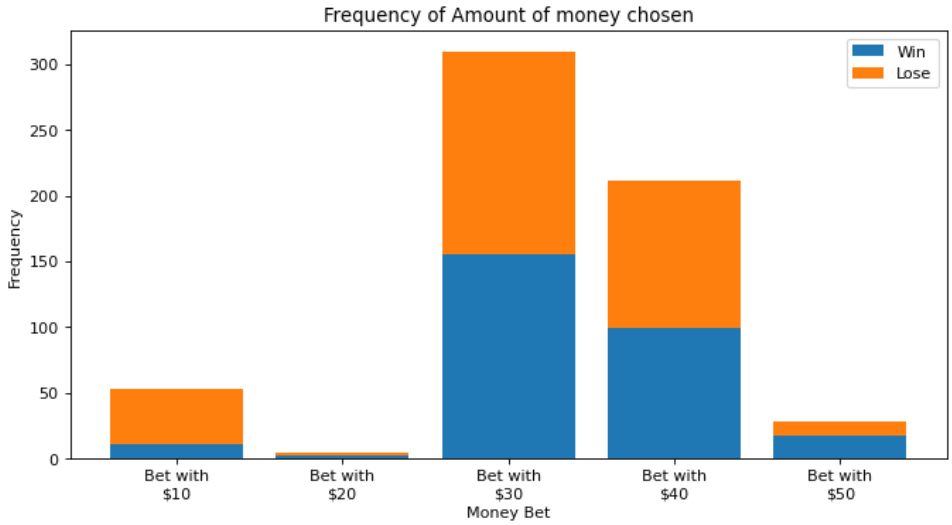

Figure 5.40 Frequency of Amount of Money Bet in Training

Figure 5.40 shows the frequency of different amounts of money bet and the win ratio of each amount of money in the testing process. Generally, we can see that the win ratio reduces for all options which matches the cash balance in figure 5.36. Among all options, the winning ratio of betting with $30 and $40 has reduced from 60.722% to 48.752% which could be the reason that the cash balance performs much worse than the model in type 1 environment.

Win Rate

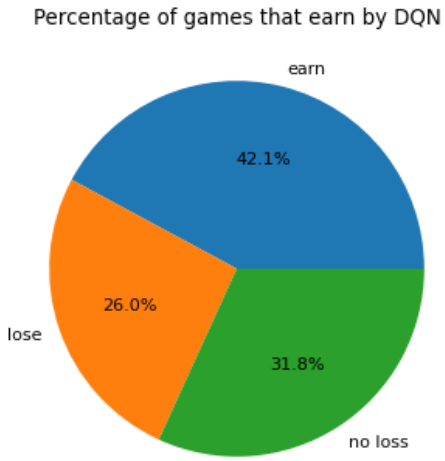

Figure 5.41 Win Rate of Agent in Training

Figure 5.41 shows the percentage of winning, losing and not betting in the training process. The agent achieves a low losing rate of 26.0% and a high winning rate of 42.1% out of 707 races in the training. Compared to figure 5.32, the percentage of not betting has significantly increased by 26% since the reward of not betting is much less compared to those betting on the horse which loses.

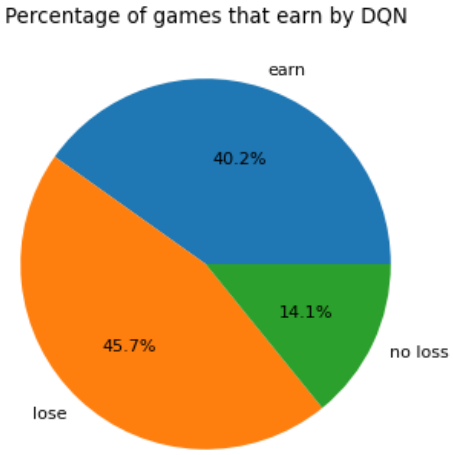

Figure 5.42 Win Rate of Agent in Testing

Similarly, figure 5.42 shows the percentage of winning, losing and not betting in the testing process which has a worse performance than the training process. Compared to figure 5.41, the agent performs worse in testing than in training where it can only achieve an acceptable winning rate of 40.2% out of 707 races in the testing and the percentage of not betting reduces by nearly 17.7%.

5.2.2.4 Summary

In summary, the overfitting problem is a huge challenge in optimizing horse betting strategy with Deep Q Network, since the states in horse betting problem (the features of horses and races) would rarely be identical among different races, which increases the difficulty of retrieving an optimal solution without overfitting the training data. In order to avoid this problem, we tried reducing the steps for training.

In our experiment, the agent tends to adapt a safer strategy on horse selection that mostly bet on horses 1 and 4. However, the low accuracy of the finishing time prediction potentially hindered the performance of the agent such that the difficulty of selecting the correct horses increases but overall, the results match with our expectations.

5.2.3 Proximal Policy Gradient

5.2.3.1 Hyperparameters

Discount Factor

The discount factor in reinforcement learning is used to discount the future rewards which reflects that the earlier action would have a less and less impact on the future action and rewards. In the previous section training the DQN agent, a γ of 1 is used which took reference from the previous project, since the number of races T is finite and the importance of both future bets and early bets are identical [16]. In contrast, in this section, we would also use 0.1 as a discount factor since the only relationship between the states in different stages of the training phase is the cash balance, thus the earlier rewards should have limited influence on the final cash balance.

Clip Range

As mentioned in section 2.5.3, a constraint with KL divergence can be used to makes sure that theta and theta' cannot differ by much in PPO. In this experiment, a clipped PPO would be adopted where the objective function is defined as follow [51]

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)\widehat{A_t}, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t}\right)\right] \quad (15)$$

such that the probability ratio term would be clipped with a clip range $[1-\epsilon, 1+\epsilon]$ where ϵ is the hyperparameter we are going to define. After several experiments and testing, we would set ϵ as 0.15 in our experiment.

Structure of Neural Network

For type 1 environment, the neural network that is used in this experiment has 2 hidden layers with 64 nodes and uses the Tanh activation function in each layer. At the input and output layer, there are 309 and 15 nodes respectively. For type 2 environment, there will be 75 output nodes and the structure of other parts remains unchanged.

5.2.3.2 Results and Analysis (Type 1)

In this section, the result of horse betting trained with Proximal Policy Optimization with the type 1 environment (fixed amount of money bet) would be discussed. Overall, the results of the testing phase performed better than the result of Deep Q Network.

Cumulative Reward

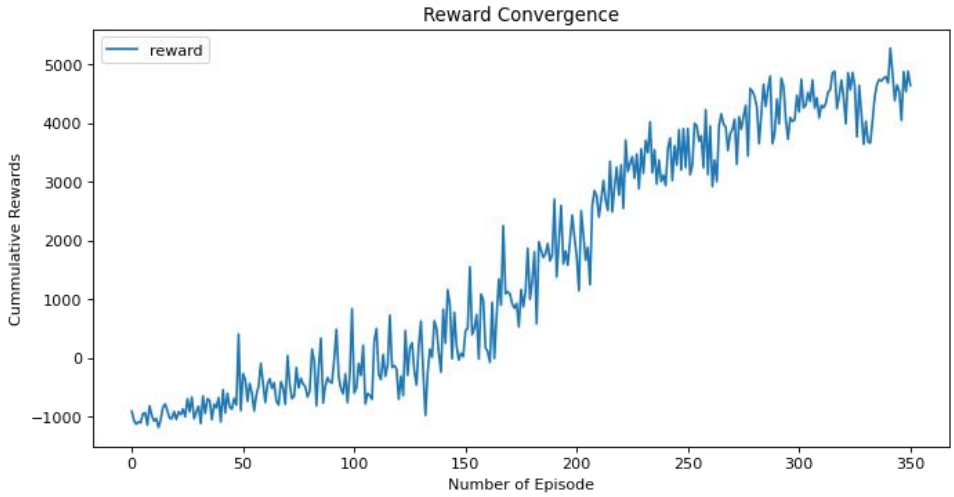

Figure 5.43 Reward Convergence of PPO with Environment 1

Figure 5.43 shows the cumulative reward per episode during the training phase. The reward is increasing and converging at the 212100th step. Similarly, the number of steps is limited in order to avoid serious overfitting.

Cash Balance

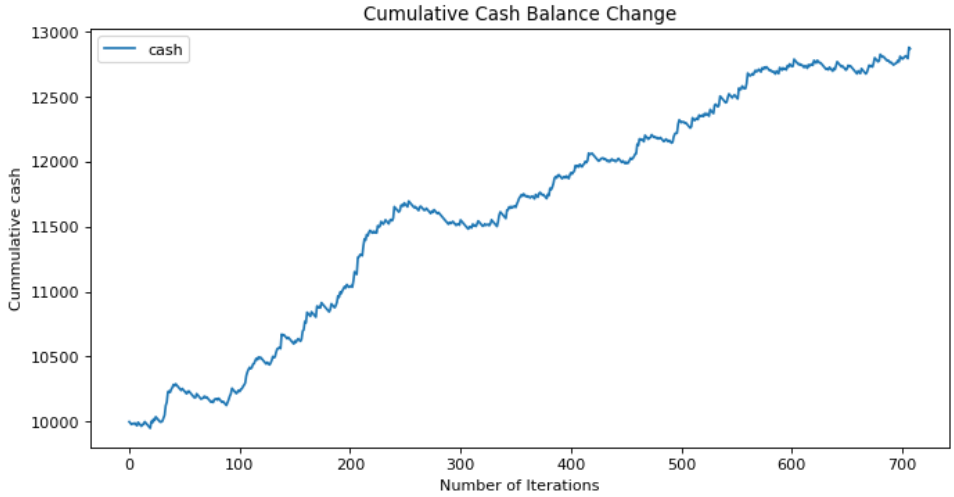

Figure 5.44 Cumulative Cash Balance Change in Training

Figure 5.44 shows the cumulative cash balance that the agent gained with bet with the training data. The agent earned $2864 after betting on 707 races. Recall that the agent trained by DQN finally earned $2027 which is 29.2% less than the current agent.

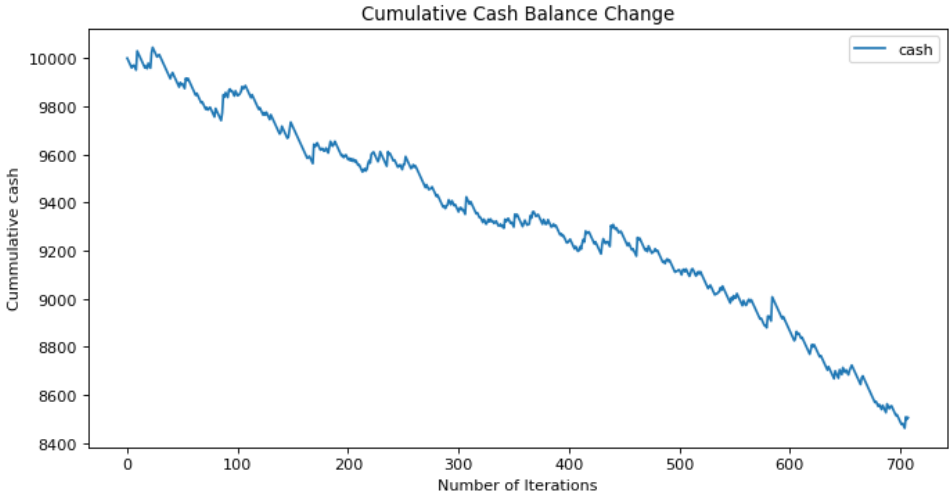

Figure 5.45 Cumulative Cash Balance Change in Testing

In contrast, figure 5.45 shows the cumulative cash balance that the agent gained with bet with the testing data. The agent achieved the highest cash balance $10045 while the cash budget gradually decreases during the testing and finally the agent has $8505 cash remaining which is 7.93% less than the agent trained with DQN

Frequency of Actions

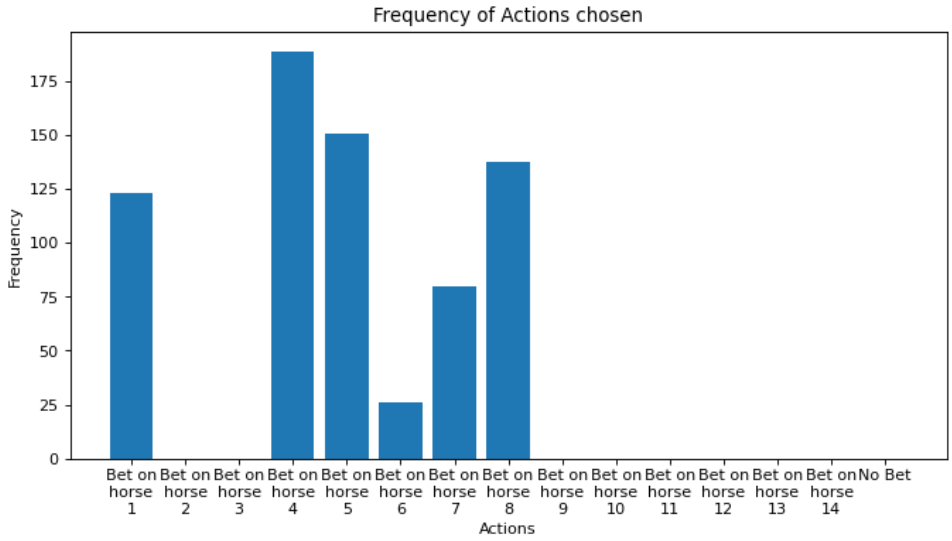

Figure 5.46 Actions Selected in Training

Figure 5.46 shows the frequency of the actions selected by the agent during the training process. Recall that the horses are ordered by their finishing time prediction from the random forest in ascending order which means that horse 1 would be predicted as the fastest horse in each race. From the graph, we can see that the agent tends to choose horses 4, 5 and 8 mostly which behaves differently and bet on horses with higher predicted finishing time compared to the previous agent trained by DQN. Although the 4th most selected action is horse 1, the options selected are mainly in the middle range of the predicted finishing times and the agent didn't choose not betting at all which can be regarded as a strategy with higher risk.

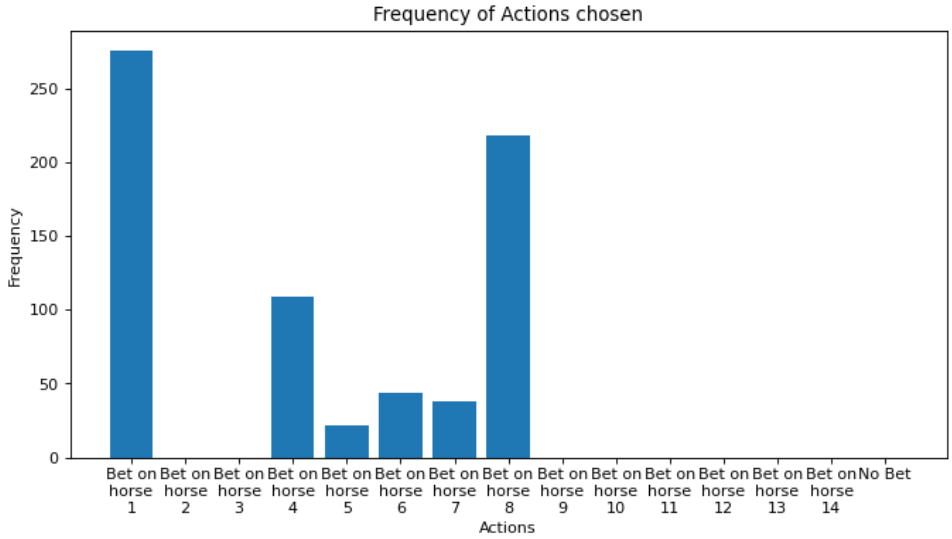

Figure 5.47 Actions Selected in Testing

However, figure 5.47 shows the frequency of the actions selected by the agent during the testing process and the most selected actions is betting on horse 1. Also, the frequency of selecting horse 8 increased slightly while the frequency of selecting horse 4, 5, and 7 greatly reduced. On the other hand, the set of actions selected in both training and testing phase are the same.

Compared to the betting strategy trained by DQN, the PPO agent placed more bets on those horses with higher finishing time predictions. Usually, those horses have a higher place odd which leads to the results of gaining more profits than the DQN agent in training part. However, as those horses bet by the PPO agent are less likely to get into top 3 places and the accuracy of the finishing time prediction is not high which leads to the result of losing more money in the testing phase.

Win Rate

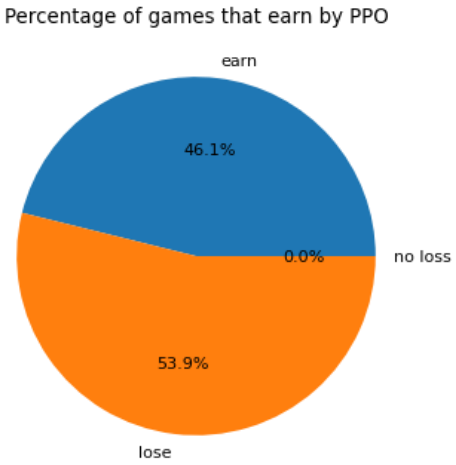

Figure 5.48 Win Rate of Agent in Training

Figure 5.48 shows the percentage of winning, losing and not betting in the training process. The PPO agent only selected the horses betting actions but never selected the non-betting actions. A high winning rate of 46.1% out of 707 races in the training is achieved which outperforms the results of the DQN agent with the same situation.

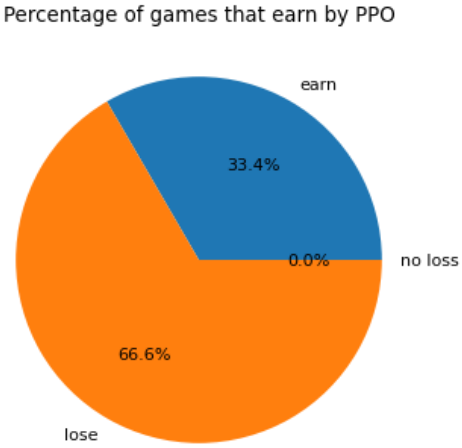

Figure 5.49 Win Rate of Agent in Testing

Figure 5.49 shows the percentage of winning, losing and not betting in the testing process. Compared to figure 5.48, the agent also performs worse in testing than in training where it can only achieve an acceptable winning rate of 33.4% out of 707 races in the testing which is 6.8% lower than the DQN agent.

5.2.3.3 Results and Analysis (Type 2)

In this section, the result of horse betting trained Proximal Policy Optimization with the type 2 environment (dynamic amount of money bet) would be discussed. Overall, the results of the testing phrase are acceptable while the agent performs worse than the previous model.

Cumulative Reward

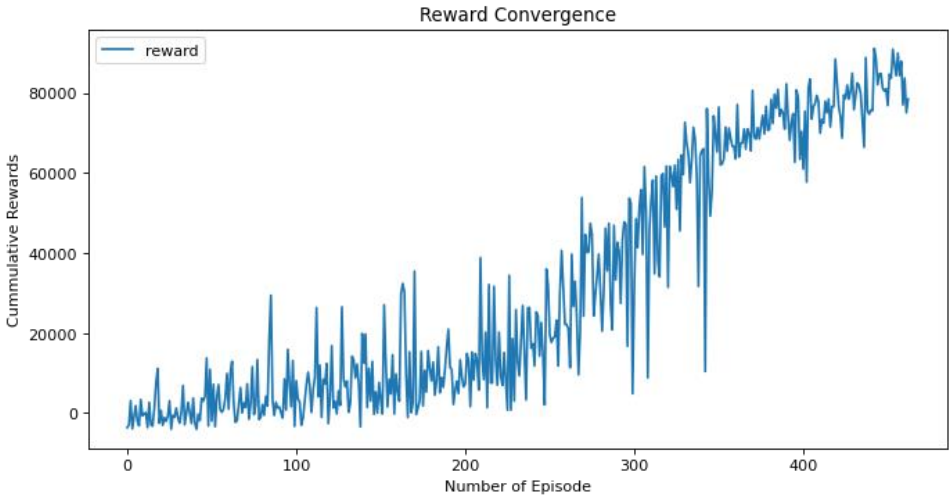

Figure 5.50 Reward Convergence of PPO with Environment 2

Figure 5.50 shows the cumulative reward convergence of PPO which is increasing and converging after 212100 steps while the variance of the rewards is greater than the curve shown in figure 5.41 but less than the curve shown in figure 5.34.

Cash Balance

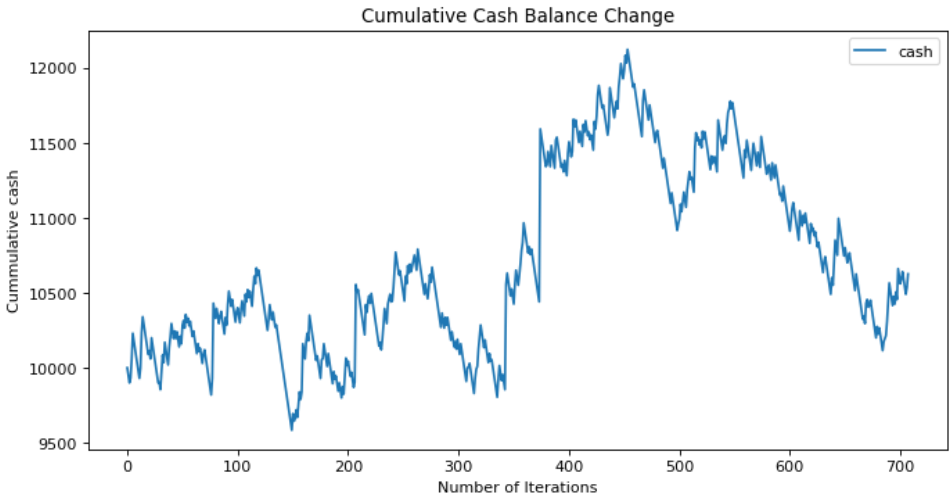

Figure 5.51 Cumulative Cash Balance Change in Training

Figure 5.51 shows the cumulative cash balance that the agent gained with bet with the training data and there exists a lot of cash change with high slope. In the early game, the cash balance dropped to $9585 while in the late game it reached the highest cash balance $12120 and finally the cash balance of the agent is $10625 after betting on 707 races. The huge fluctuation shows a low stability of gaining profits.

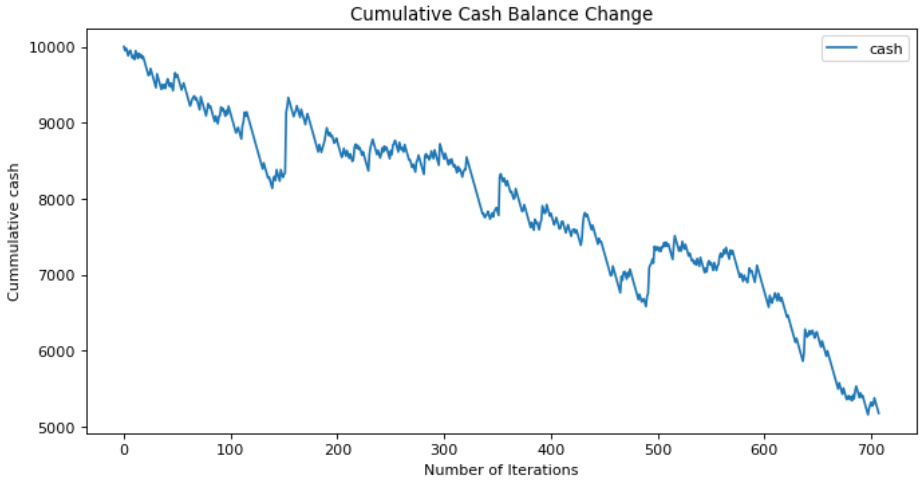

Figure 5.52 Cumulative Cash Balance Change in Testing

Similarly, the performance in the testing phase is worse than the result of the training phase and figure 5.52 shows the cash balance of PPO agent in the testing phase. The performance is unsatisfying since the final cash balance of the agent is only $5175, which is equivalent to a 50% loss. The reason for this is mainly because the agent chooses the betting with the highest amount of money in every race, which would be shown in later sections.

Frequency of Actions

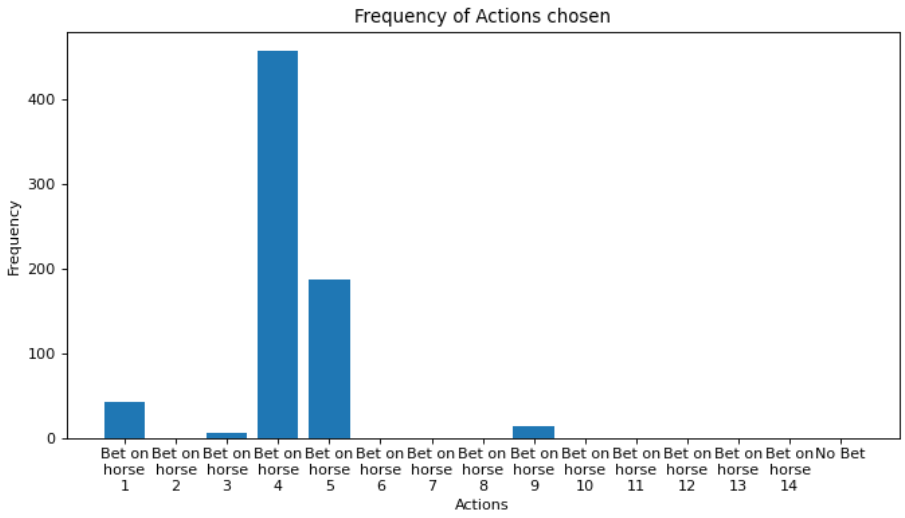

Figure 5.53 Actions Selected in Training

Figure 5.53 shows the frequency of the actions selected by the agent during the training process. Recall that the horses are ordered by their finishing time prediction from the random forest in ascending order which means that horse 1 would be predicted as the fastest horse in each race. The agent prefers selecting horse 4 and 5 rather than other options which is different from the PPO agent trained with type 1 environment. However, both PPO agents are not trying to pick the option of not betting which would potentially lead to a huge loss.

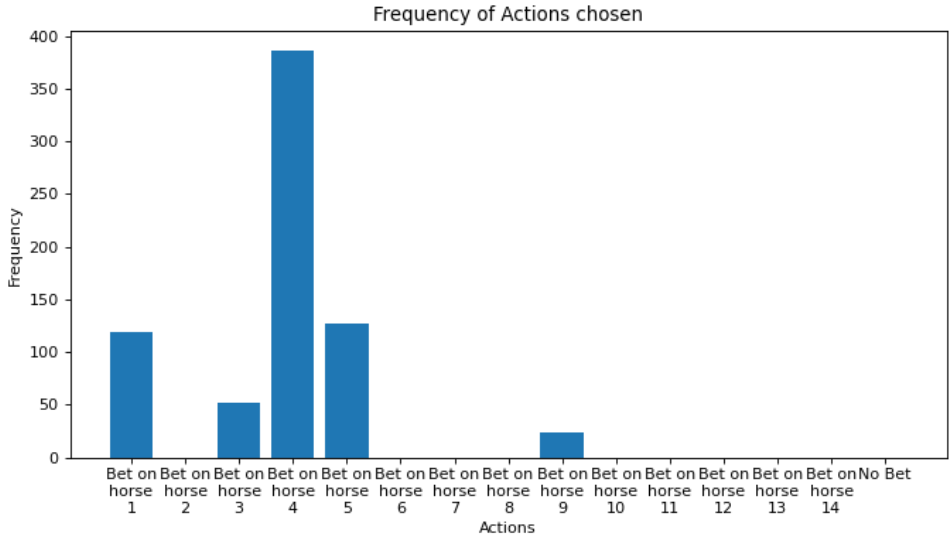

Figure 5.54 Actions Selected in Testing

Figure 5.54 shows the frequency of the actions selected by the agent during the testing process. The main difference between actions selected in training and testing is that horse 4 and 5 has been less likely chosen, and the frequency of choosing horses 1, 3, 5 and 9 has risen on a small scale.

Compared to the PPO agents across the type of environment, both of the agents do prefer betting in every race but instead, they place their bets on the horses which have a predicted finishing time in the middle range mainly.

Amount of Money Bet

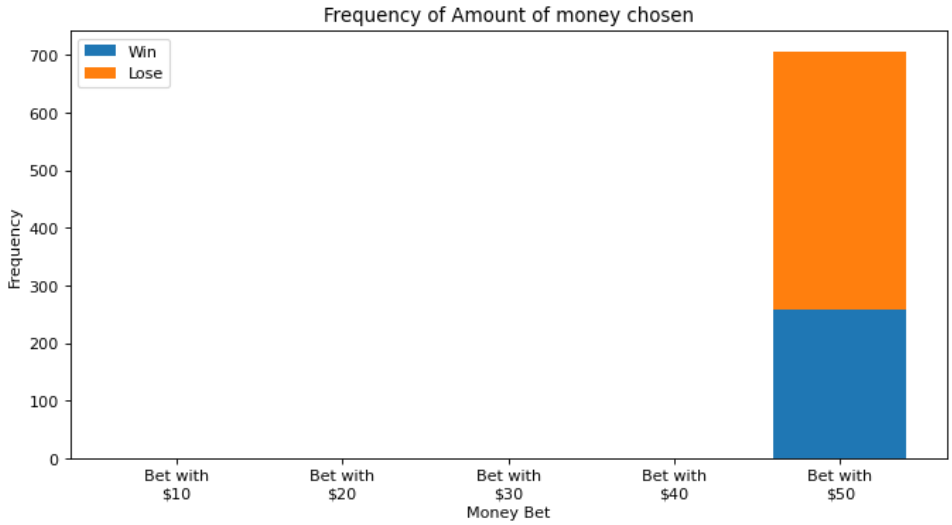

Figure 5.55 Frequency of Amount of Money Bet in Training

Figure 5.55 shows the frequency of different amounts of money bet and the win ratio of each amount of money in the training process. From the graph we can see that the agent only bet $50 on all horses, which would be the main reason that many steep slopes shown in figure 5.51. Also, the agent cannot achieve our target that deciding different amounts of money to bet on different horses based on the states.

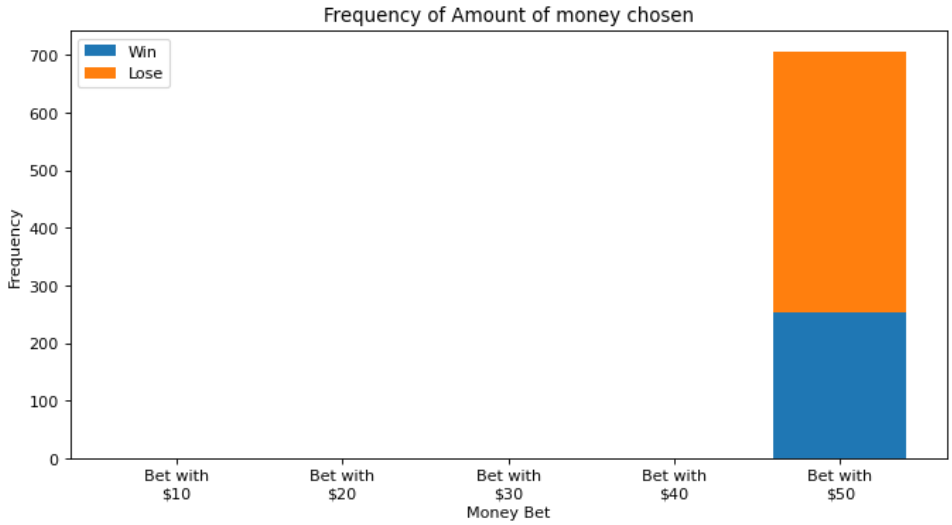

Figure 5.56 Frequency of Amount of Money Bet in Testing

Figure 5.56 shows the frequency of different amounts of money bet and the win ratio of each amount of money in the testing process. Since the agent is trained to bet $50 on all horses no matter

the state of the races and the low accuracy of finishing time prediction, they lead to the huge loss shown in figure 5.52.

Win Rate

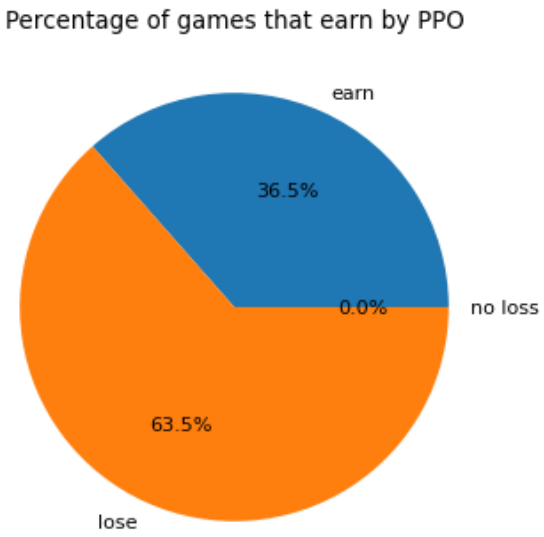

Figure 5.57 Win Rate of Agent in Training

Figure 5.57 shows the percentage of winning, losing and not betting in the training process. The agent achieves an acceptable winning rate of 36.5% which is lower than the result of DQN by 5.6% in the same environment.

Figure 5.58 Win Rate of Agent in Testing

Similarly, figure 5.58 shows the percentage of winning, losing and not betting in the testing process which has a slightly worse performance than the training process. The winning rate reduced by 0.6% only which shows that the huge loss shown in figure 5.52 is mainly affected by the amount of money betting.

5.2.3.4 Summary

In summary, the overfitting problem is also a challenging problem in optimizing horse betting strategy with Proximal Policy Optimization, which is similar to Deep Q Network. It can be shown in section 5.3.3.3 where the cash balance in the training phase is significantly greater than the cash balance in the testing phase. In contrast, the difference in win rate of both is only 0.6% the difference in cash balance has reached more than $5000.

On the other hand, in our experiment, the agent tends to take a horse betting strategy which has a higher risk than the DQN agent learned on horse selection. The PPO agent prefers betting the horses that mostly has a predicted finishing time within the middle range while the probability of those getting into the top 3 places is relatively low. Especially when the agent is trained in type 2 environment and betting $50 on every race, the effects of this horse betting strategy are reflected in the cash balance.

5.2.4 Augmented Random Search

5.2.4.1 Hyperparameters

Standard Deviation of the Exploration Noise

The exploration noise v is a constant positive value, where $v < 1$, and applied during the sampling procedure in the Augmented Random Search algorithm. If the value of v is smaller, the results would be closer to the original objectives. In contrast, if the value of v is too large which may potentially harm the performance of the algorithm as the sampled results may have a large variance from the actual objective. In our experiment, we set 0.05 as the standard deviation of exploration noise since we prefer selecting a smaller v to avoid large variance problem.

Number of Directions Sampled and Top-performing Directions

As mentioned in section 2.5.4, the number of top-performing directions b is used during the step update procedure. Only the top b rewards would be selected after sorting them in descending order in order to maximizing the collected rewards and improve the result of the algorithm, in our experiment, we set b as 16 and the number of directions sampled as 20.

Structure of Neural Network

For type 1 environment, the neural network that is used in this experiment has 2 hidden layers with 64 nodes and uses the ReLU activation function in each layer. At the input and output layer, there

are 309 and 15 nodes respectively. For type 2 environment, there will be 75 output nodes and the structure of other parts remains unchanged.

5.2.4.2 Results and Analysis (Type 1)

In this section, the result of horse betting trained with Augmented Random Search with the type 1 environment (fixed amount of money bet) would be discussed. Overall, we have tried applying different combinations of the hyperparameters while the results of the testing phase do not achieve our expectations.

Cumulative Reward

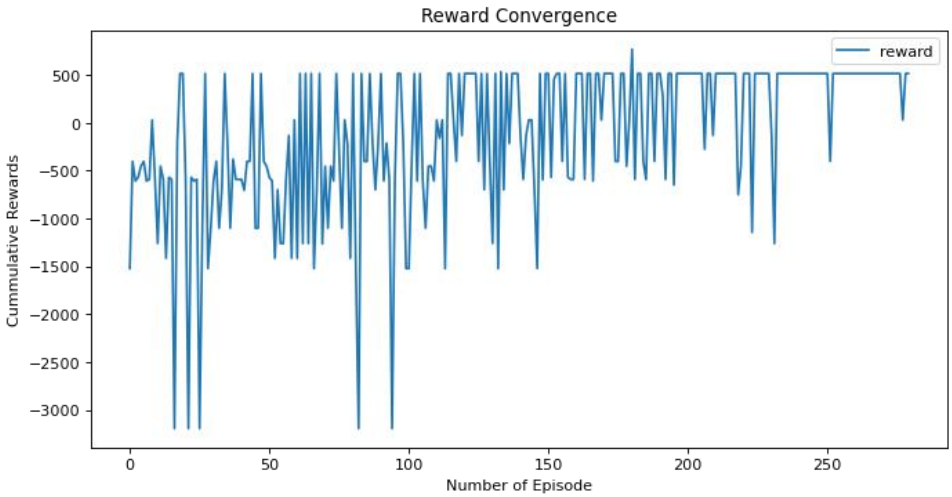

Figure 5.59 Reward Convergence of ARS with Environment 1

Figure 5.59 shows the cumulative reward per episode during the training phase. There exist many fluctuations and the average of the rewards is increasing with the number of episodes trained. It is obvious to see that there is a boundary limiting the maximum of the cumulative rewards and the reward tends to converge at 512.5.

Cash Balance

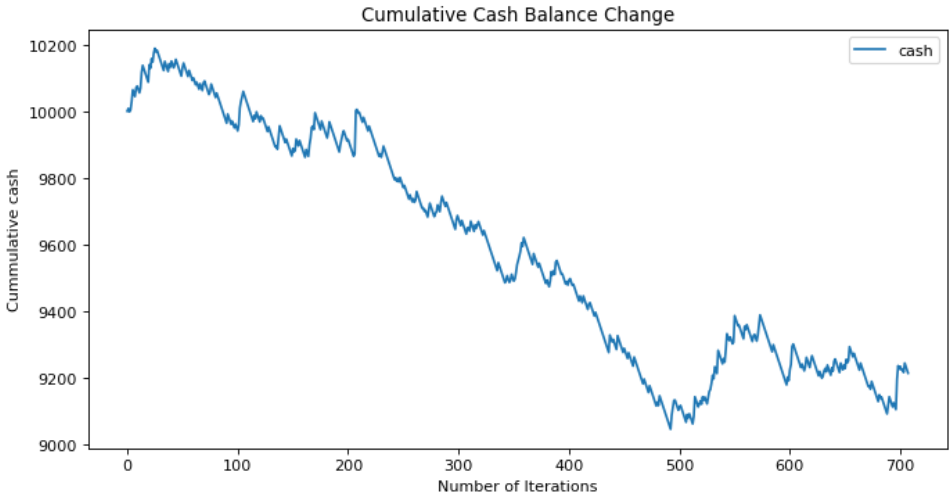

Figure 5.60 Cumulative Cash Balance Change in Training

Figure 5.60 shows the cumulative cash balance that the agent has after completing the training phase. The agent used to have $9047 after the 500th race while it lost $785 after betting on 707 races which is the worst result temporarily.

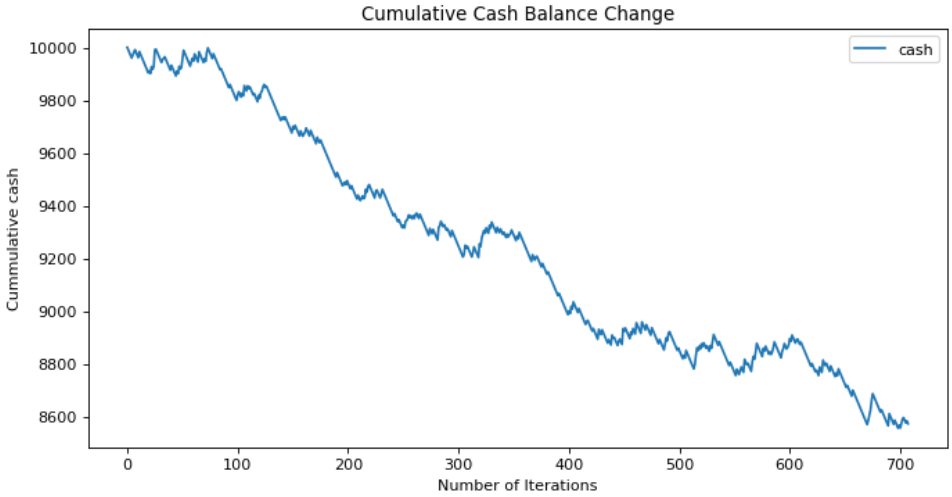

Figure 5.61 Cumulative Cash Balance Change in Testing

In contrast, figure 5.61 shows the cumulative cash balance that the agent gained with bet with the testing data. Recall that the net cash change of the agent in the training phase is negative. In the

testing phase, the agent also has a negative net change and lost $1428 which is surprisingly slightly less than the agent lost in PPO testing.

Frequency of Actions

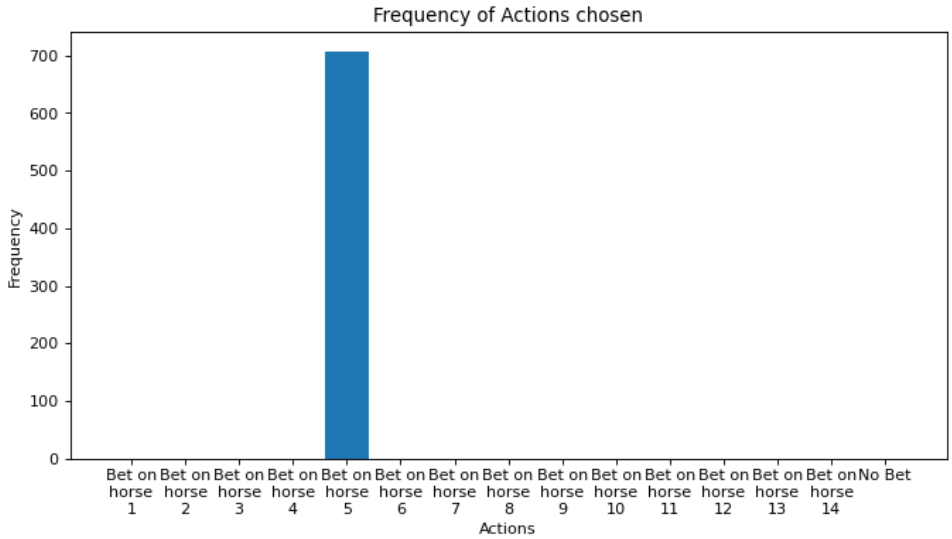

Figure 5.62 Actions Selected in Training

Figure 5.62 shows the frequency of the actions selected by the agent during the training process. Recall that the horses are ordered by their finishing time prediction from the random forest in ascending order which means that horse 1 would be predicted as the fastest horse in each race. From the graph, we can see that the agent only chooses horse 5, which is a betting strategy with a middle level of risk. It can be a reason why the agent performs badly in the training phase.

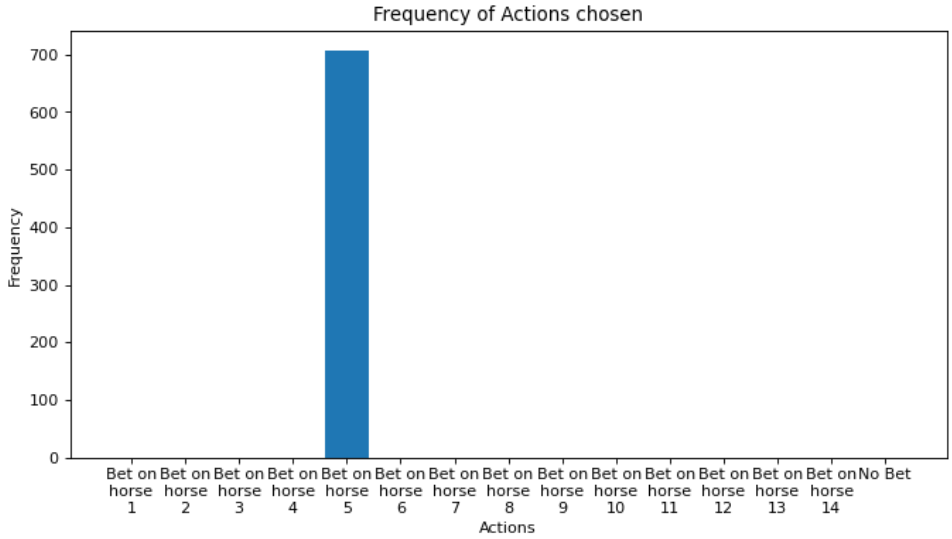

Figure 5.63 Actions Selected in Testing

Similarly, figure 5.63 shows the frequency of the actions selected by the agent during the testing process and the most selected actions is also betting on horse 5 which shows that the agent is not able to select different horse to bet based on the situation of the race.

Compared to the performance of the agents trained by DQN and PPO, they can be trained to choose different actions which are regarded as a safer strategy while the ARS agent bet all its money on horse 5 every race and horse 5 is in the middle rank ordered by the ascending finishing time prediction. Although the money gain would be greater if horse 5 won, the probability of horse 5 getting a win is low which generally leads to an unacceptable performance.

Win Rate

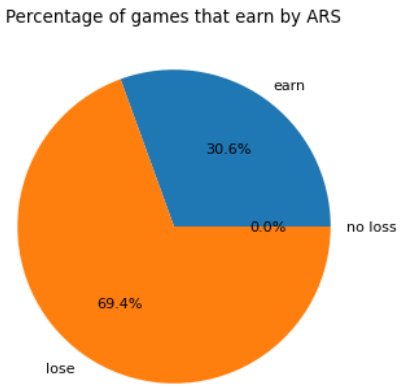

Figure 5.64 Win Rate of Agent in Training

Figure 5.64 shows the percentage of winning, losing and not betting in the training process. Similar to the PPO agent, the ARS agent never selects the non-betting option over the 707 races. It recalled that the ARS selected horse 5 for all 707 races and it achieved a 30.6% winning rate. The percentage is less than both DQN agent and PPO agent by 26.8% and 15.5% respectively.

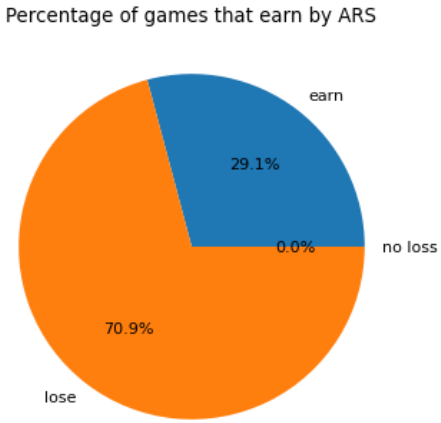

Figure 5.65 Win Rate of Agent in Testing

Figure 5.65 shows the percentage of winning, losing and not betting in the testing process. Compared to figure 5.64, the agent also performs slightly worse in testing than in training where it can only achieve a winning rate of 29.1% out of 707 races in the testing which is 1.5% lower than the agent in training phase.

5.2.4.3 Results and Analysis (Type 2)

In this section, the result of horse betting trained Proximal Policy Optimization with the type 2 environment (dynamic amount of money bet) would be discussed. Overall, the results of the testing phrase are acceptable while the agent performs worse than the previous model.

Cumulative Reward

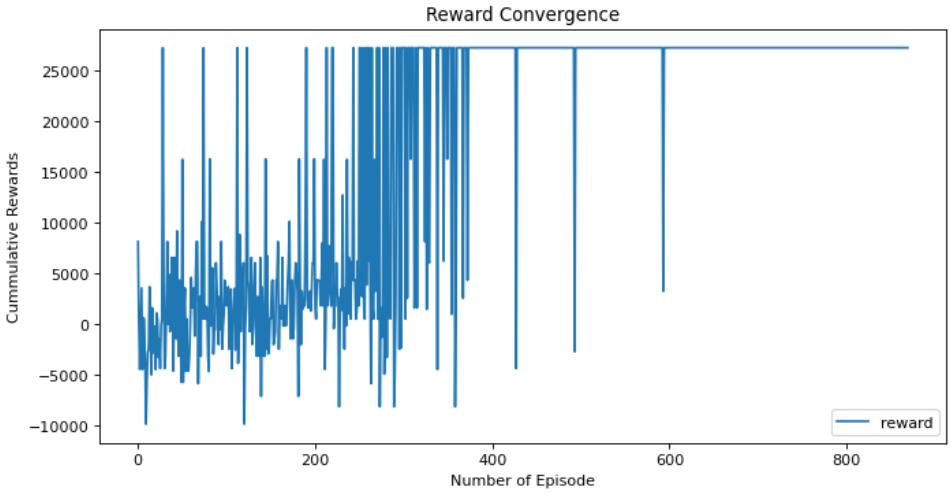

Figure 5.66 Reward Convergence of ARS with Environment 2

Figure 5.66 shows the cumulative reward convergence of ARS which is increasing and converging after 140000 steps. Similar to figure 5.59 there exists an upper limit that bounds the cumulative rewards and fluctuations of rewards as well.

Cash Balance

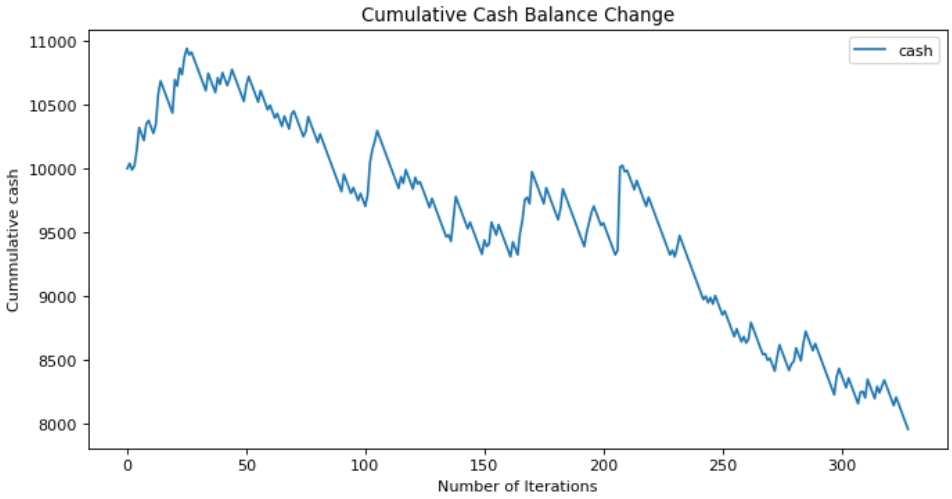

Figure 5.67 Cumulative Cash Balance Change in Training

Figure 5.67 shows the cumulative cash balance that the agent gained by betting with the training data and the highest cash balance the agent has is $10940 and finally the cash balance decreases to $7960 after betting on all 329 races. Since the lower limit of the cash balance is set as $8000 and the agent is not able to keep the cash balance greater than the limit and the training stops at 329 races. This is also a problem that we face when we are trying to optimize a horse betting strategy using ARS.

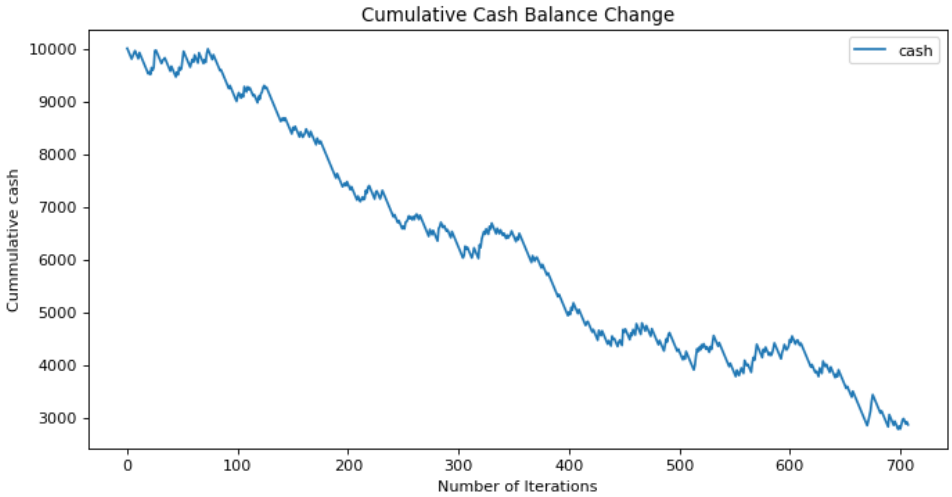

Figure 5.68 Cumulative Cash Balance Change in Testing

Similarly, the performance of the agent in the testing phase is unsatisfactory where the cash balance after betting on 707 races is only $2860 equivalent to a 71,4% loss as shown in figure 5.68. The

main reason is that the agent only selects betting on every horse with $50 which would be shown in the later section.

Frequency of Actions

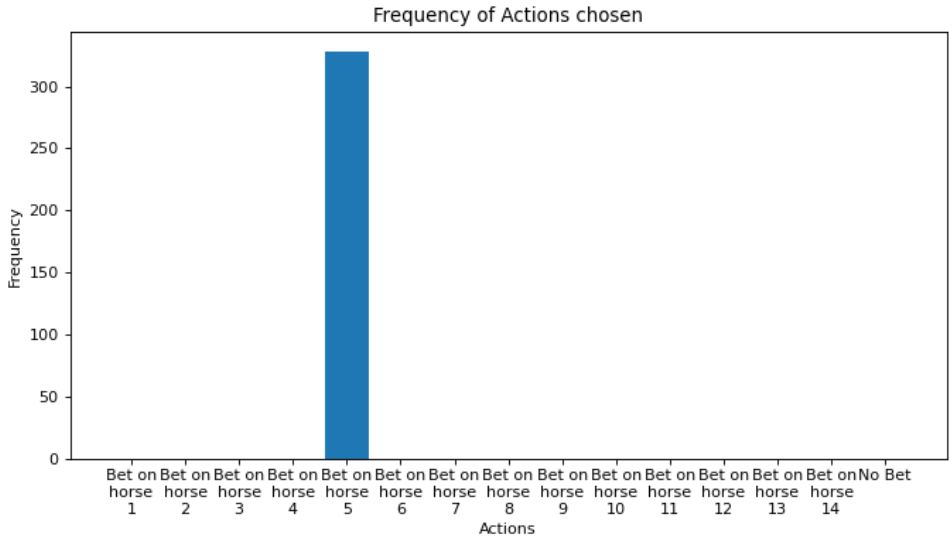

Figure 5.68 Actions Selected in Training

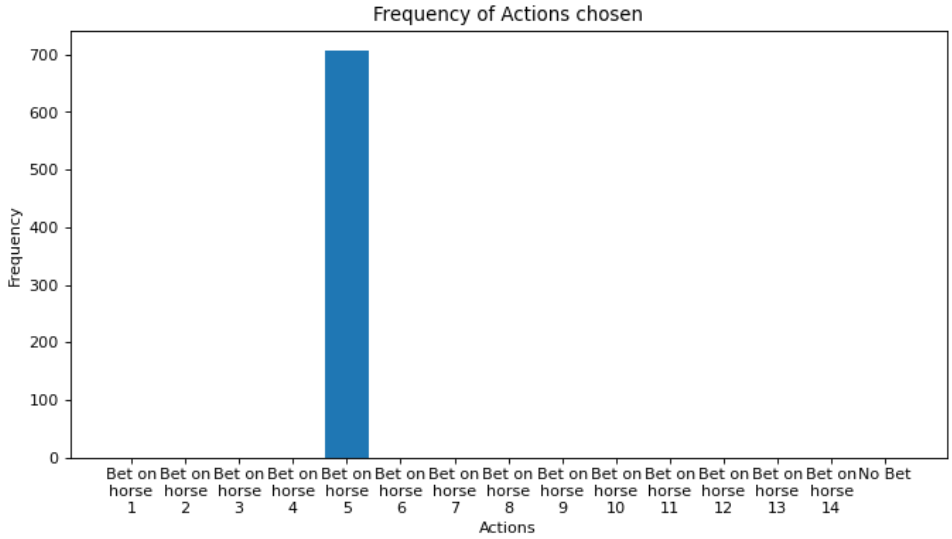

Figure 5.69 Actions Selected in Testing

Similar to the behavior of the ARS agent trained in the type 1 environment, the new agent also selects horse 5 every race in both training and testing as shown in figures 5.68 and 5.69 respectively.

Amount of Money Bet

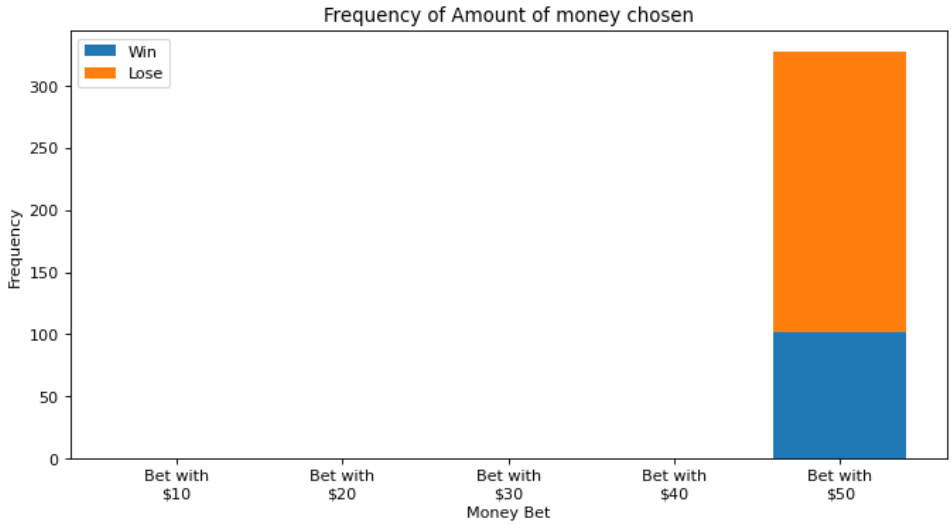

Figure 5.70 Frequency of Amount of Money Bet in Training

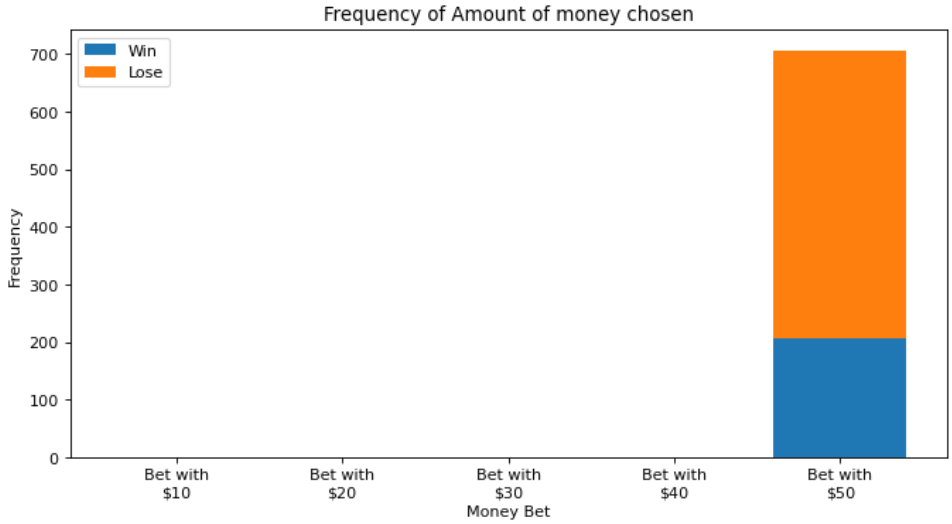

Figure 5.71 Frequency of Amount of Money Bet in Testing

Figure 5.70 and 5.71 shows that the ARS agent selects the option of betting with $50 every race which is the major reason for losing money and the incompleteness of training phase. Since horse 5 is not predicted to win with a high probability as well as the place odd is not high enough to recover the loss with a few wins only.

Win Rate

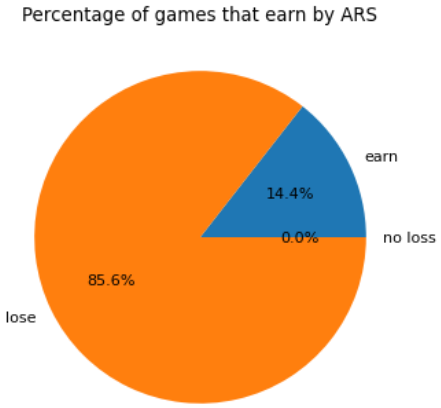

Figure 5.72 Win Rate of Agent in Training

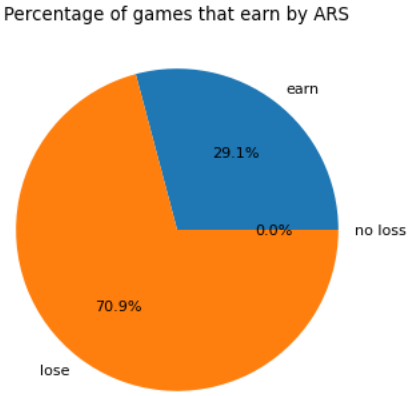

Figure 5.73 Win Rate of Agent in Testing

Similarly, figures 5.72 and 5.73 are supposed to be identical to the figures 5.64 and 5.65 since both agents select horse 5 in every race while the new agent is not able to complete all races in the training phase, thus the win ratio shown in 5.72 is not the same as before.

5.2.4.4 Summary

In summary, overfitting seems not to be the most important problem when we try to apply Augmented Random Search to optimize horse betting strategy. Instead, the agent is not able to select different options or explore different options once they selected an action which is an early convergence problem shown in figure 5.59 and 5.66. In order to solve the problem, we have tried to tune the learning rate, standard deviation of the exploration noise and the number of top-performing direction while the result of lacking exploration is still there, but just the option selected are swapped to another horse (horse 1 or 8) throughout the game. Similarly, the problem also exists in selecting the amount of money bet which leads to the unsatisfied results in this experiment.

5.2.5 Cross Entropy Method

5.2.5.1 Hyperparameters

The Number of Samples and Elite Fraction

The key procedure of Cross Entropy Method is updating the distribution using the n sampled weight vector and the corresponding values obtained. As mentioned in section 2.5.5, the elite fraction p is used to define how many percentages of the sampled vectors are used to update the distribution which has a similar concept with the number of top-performing directions used in ARS. In our experiment, we set n as 20 and p as 0.8.

Noise

As mentioned in section 2.5.5, the applicability of CEM is restricted since there is a problem of early convergence which has been proved. In order to solve this problem, a noise Z would be added to the distribution during the updating procedure. However, in our experiment we set Z as 0 since the cumulative reward cannot converge no matter how many steps are taken.

Structure of Neural Network

For type 1 environment, the neural network that is used in this experiment has 2 hidden layers with 25 nodes and uses the Tanh activation function in each layer. At the input and output layer, there are 309 and 15 nodes respectively.

For type 2 environment, the neural network that is used in this experiment has 2 hidden layers with 150 nodes and uses the Tanh activation function in each layer. At the input and output layer, there are 309 and 75 nodes respectively.

5.2.5.2 Results and Analysis (Type 1)

In this section, the result of horse betting trained with Cross Entropy Method with the type 1 environment (fixed amount of money bet) would be discussed. Although the cumulative reward does not converge to any point while the result is comparable to the result of Deep Q Network and Proximal Policy Optimization.

Cumulative Reward

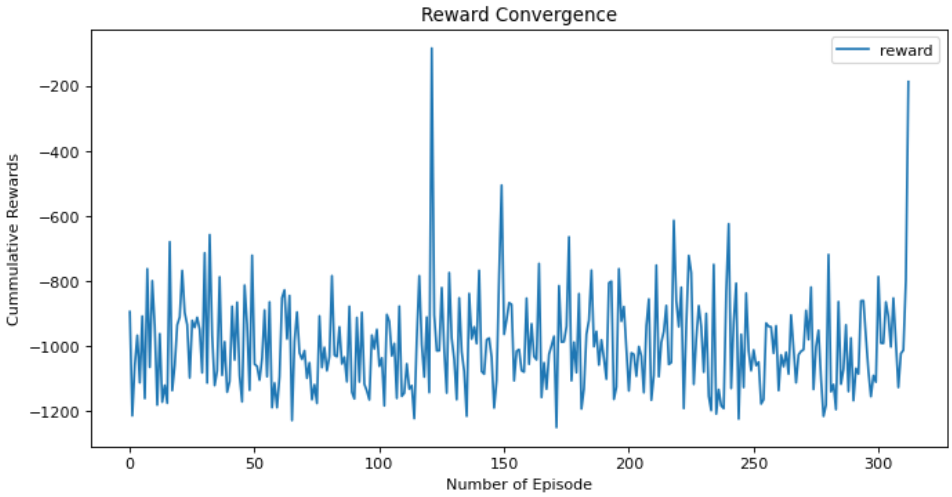

Figure 5.74 Reward Convergence of CEM with Environment 1

Figure 5.74 shows the cumulative reward per episode during the training phase after training for 100000 steps. It is obvious to see that the rewards do not tend to converge at any point no matter if no noise is added.

Cash Balance

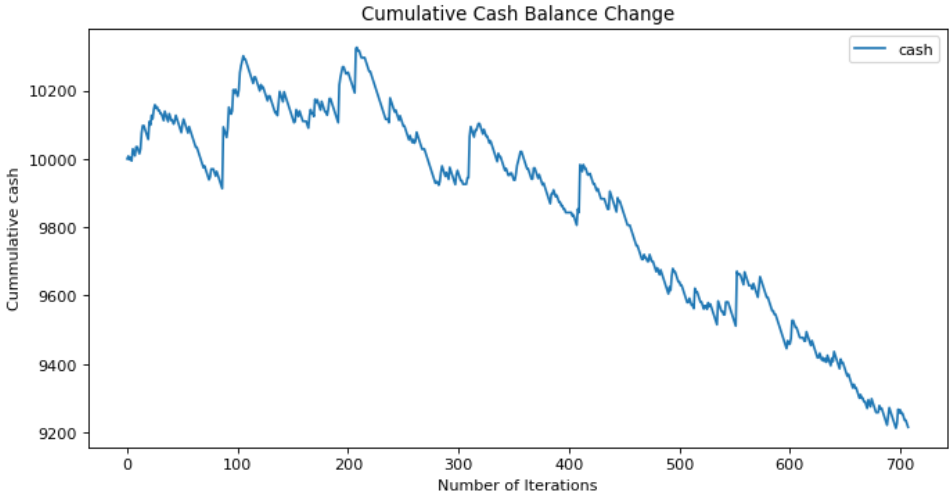

Figure 5.75 Cumulative Cash Balance Change in Training

Figure 5.75 shows the cumulative cash balance that the agent gained with bet with the training data. Although the agent gains at most $326 in the early game, the agent lost $785 after betting on 707 races which have the exact same cash balance as the ARS agent.

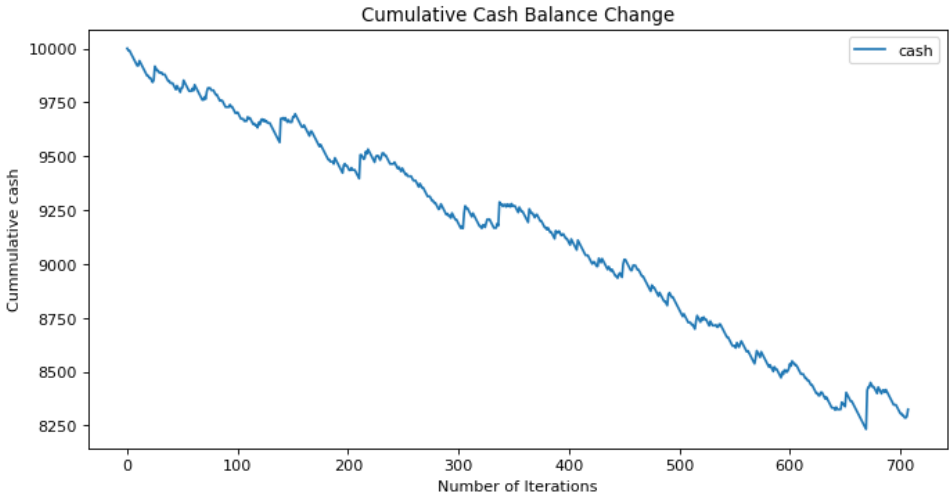

Figure 5.76 Cumulative Cash Balance Change in Testing

In contrast, figure 5.76 shows the cumulative cash balance that the agent gained with bet with the testing data and the agent finally has only $8325 after 707 races. This is the worst performance in terms of the cash balance compared across the same environment.

Frequency of Actions

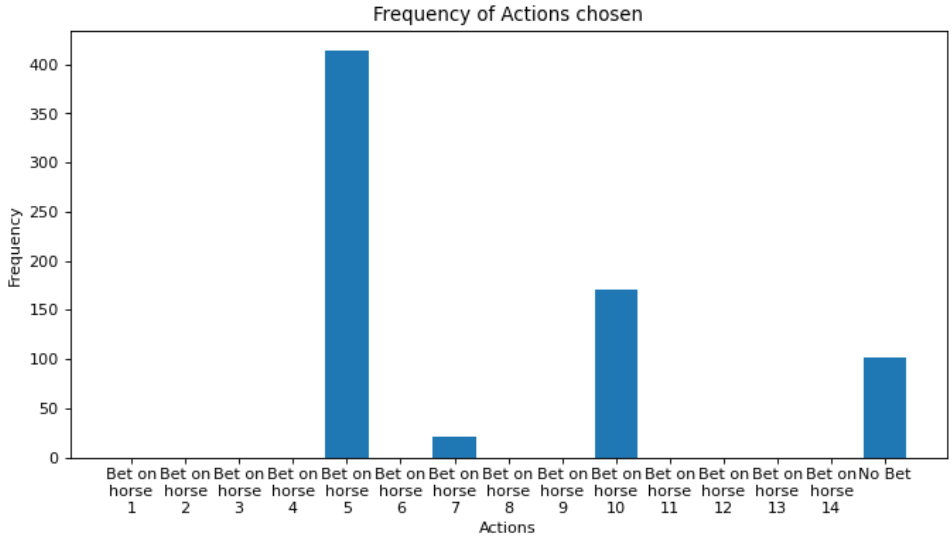

Figure 5.77 Actions Selected in Training

Figure 5.77 shows the frequency of the actions selected by the agent during the training process. Recall that the horses are ordered by their finishing time prediction from the random forest in

ascending order which means that horse 1 would be predicted as the fastest horse in each race. The CEM agent prefers to select horse 5, 10 and not bet where each option can be classified into middle risk, high risk and no risk respectively. Compared to the previous results, the CEM is able to choose diversified actions in each race while there is no evidence showing that it chooses based on the states.

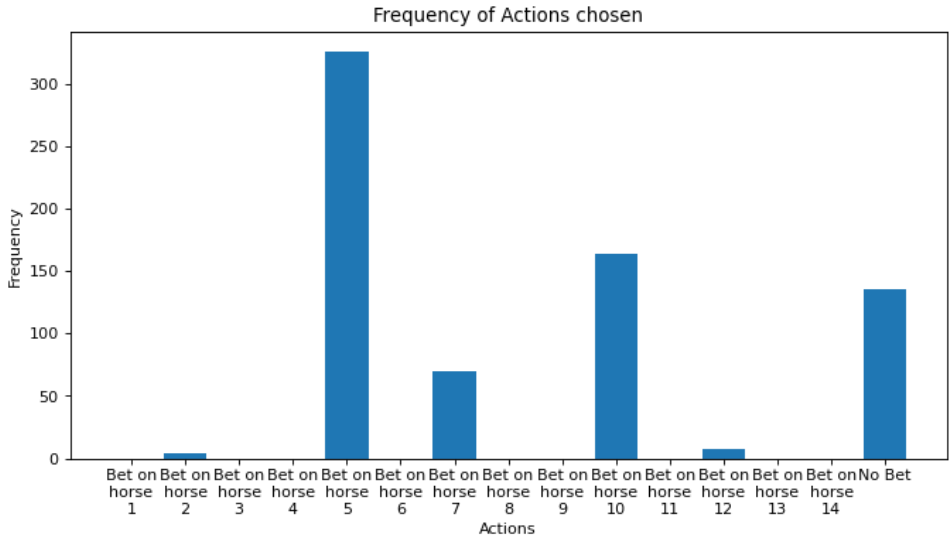

Figure 5.78 Actions Selected in Testing

However, figure 5.78 shows the frequency of the actions selected by the agent during the testing process. The frequency of selecting horse 4 decreased slightly while the frequency of selecting horse 2, 7, and 12 significantly increased. Among all the actions selected, horse 2 and horse 12 are the new actions that have not been selected in the training phase.

Win Rate

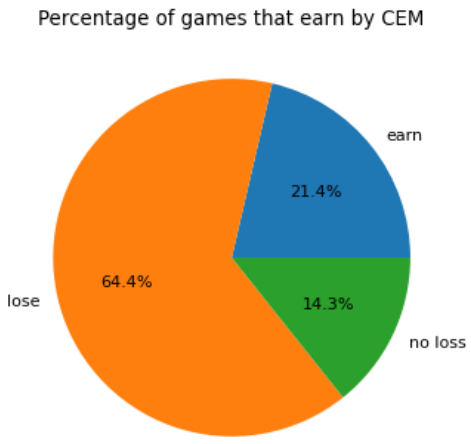

Figure 5.79 Win Rate of Agent in Training

Figure 5.79 shows the percentage of winning, losing and not betting in the training process. The ARS agent achieves a winning rate of 21.4% out of 707 races in the training which is the worst performance among all agents in the same situation. However, the ARS agent also selected 14.3% of no bet actions which has kept an acceptable cash balance shown in previous figures.

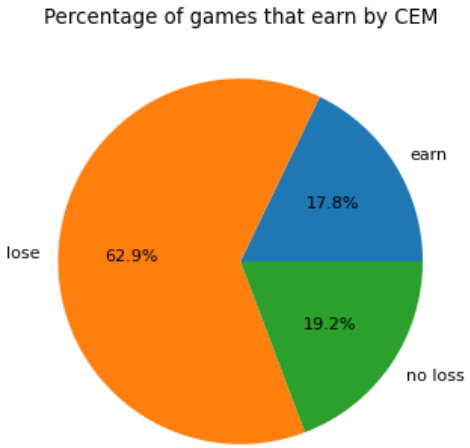

Figure 5.80 Win Rate of Agent in Testing

Figure 5.80 shows the percentage of winning, losing and not betting in the testing process. Similarly, the ARS agent also performs worse in testing than in training where it can only achieve a winning rate of 17.8% out of 707 races in the testing which is 3.6% lower than the training phase. However, the loss rate is less than before by 1.5% which shows that the agent tends to not bet in the testing phase.

5.2.5.3 Results and Analysis (Type 2)

In this section, the result of horse betting trained Proximal Policy Optimization with the type 2 environment (dynamic amount of money bet) would be discussed. The ARS agent is the only one that performs better in the testing phase in terms of profitability.

Cumulative Reward

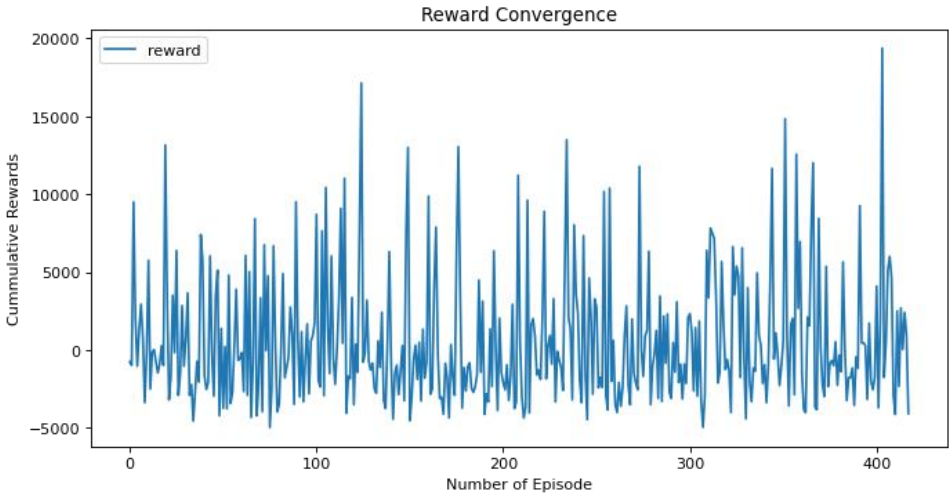

Figure 5.81 Reward Convergence of ARS with Environment 2

Figure 5.81 shows cumulative reward convergence of ARS similar to 5.64 where the cumulative reward does not tend to converge throughout the game after 100000 steps.

Cash Balance

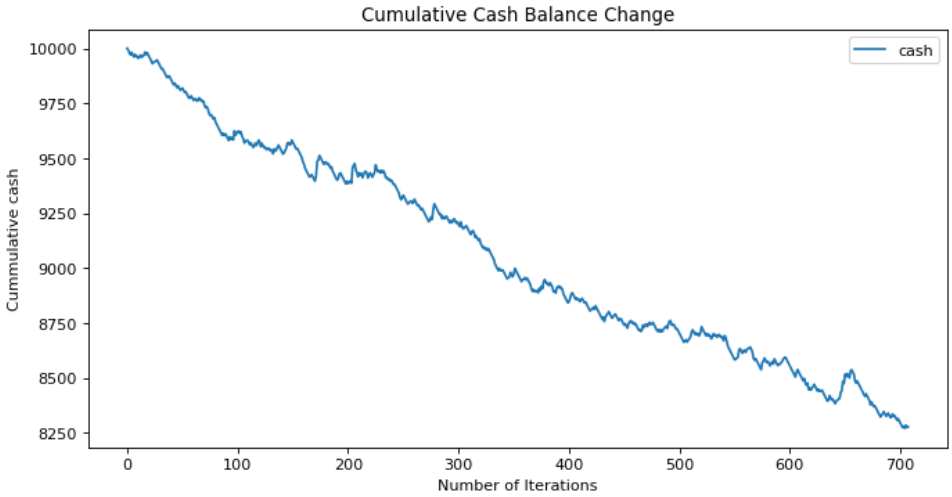

Figure 5.82 Cumulative Cash Balance Change in Training

Figure 5.82 shows the cumulative cash balance that the agent gained with bet with the training data and the agent loses money gradually. The agent has a final cash balance of $8277, which is slightly more than the ARS in the same environment by $317.

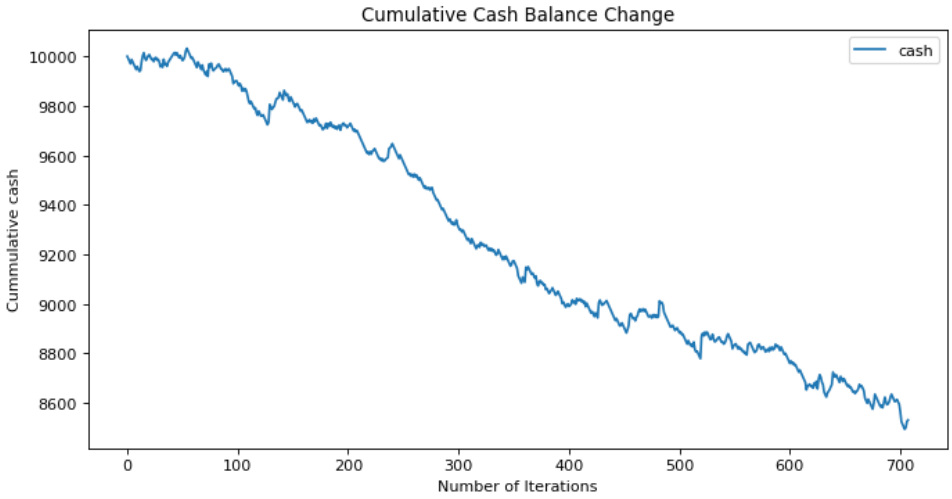

Figure 5.83 Cumulative Cash Balance Change in Testing

Figure 5.83 shows the cash balance of CEM agent in the testing phase which is the only agent that has more cash balance in the testing phase than the training phase. The CEM agent has a final cash balance of $8531 which is more than the cash balance in training phase by 3.06%.

Frequency of Actions

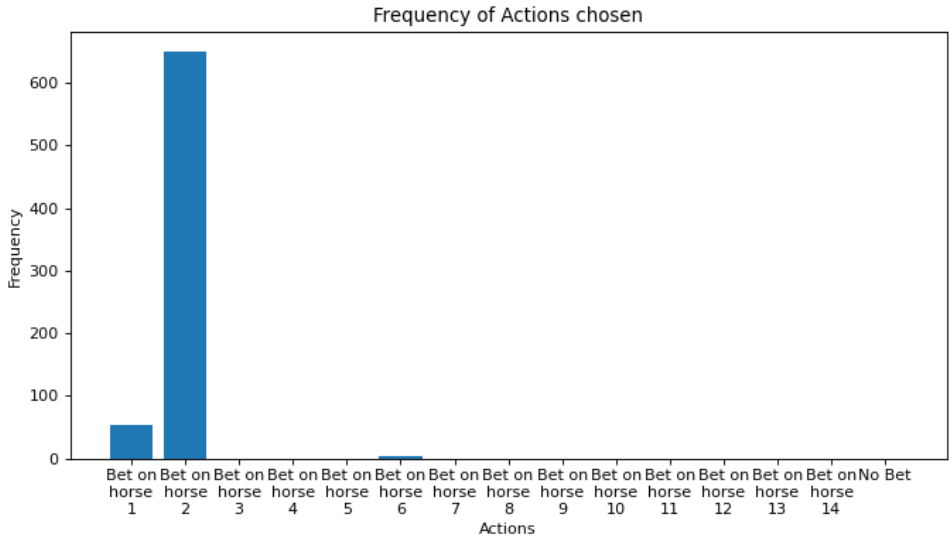

Figure 5.84 Actions Selected in Training

Figure 5.84 shows that the CEM agent selects horse 2 for most of the time. Recall that the horses are ordered by their finishing time prediction from the random forest in ascending order which means that horse 1 would be predicted as the fastest horse in each race. Other than horse 2, horse

and horse 6 also occupy a small portion of total actions and all the horses selected by the CEM are considered a low-risk preference.

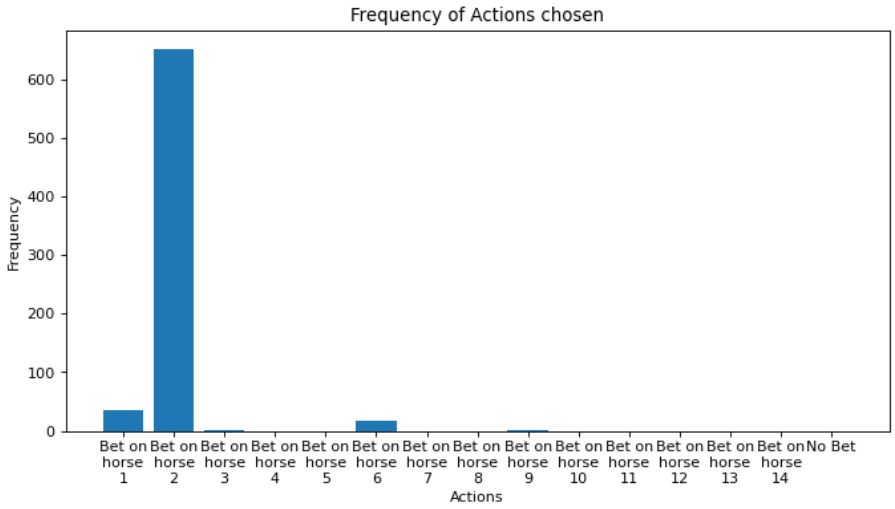

Figure 5.85 Actions Selected in Testing

Figure 5.85 shows the frequency of the actions selected by the agent during the testing process. In general, there is no significant difference between figures 5.84 and 5.85 while the frequency of selecting horse 1 and 6 increased slightly. Also, horses 3 and 9 are newly added to the selected actions but the frequency is extremely low.

Compared to the previous agents with the same type of environment, both ARS and CEM agents do prefer betting on single horse in every race but the risk of CEM agent taking is less than the ARS.

Amount of Money Bet

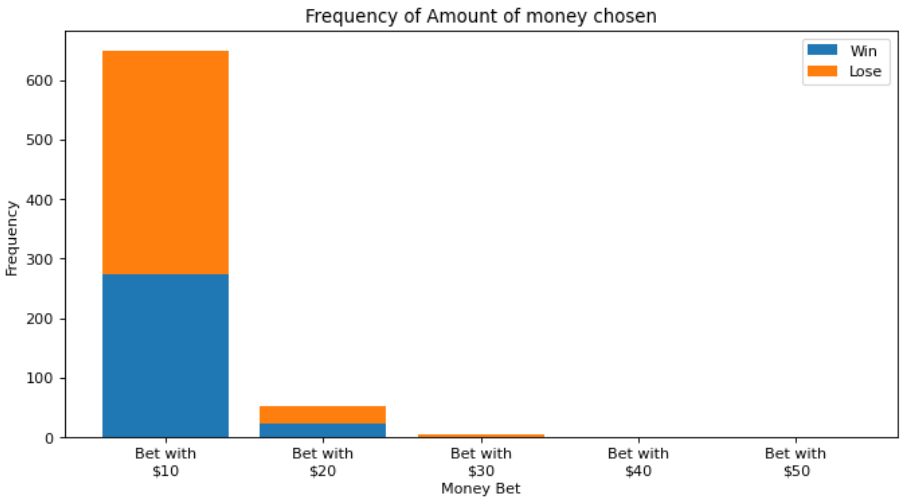

Figure 5.86 Frequency of Amount of Money Bet in Training

Figure 5.86 shows the frequency of different amounts of money bet and the win ratio of each amount of money in the training process. From the graph we can see that the agent only bet with relatively low amount of money, which would be the main reason that the profitability in testing phase is better than training phase shown in figure 5.83.

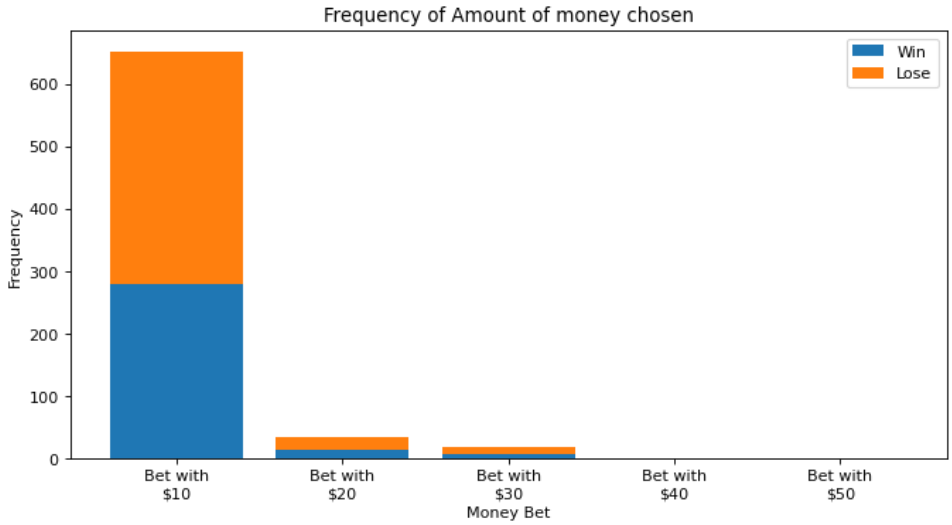

Figure 5.87 Frequency of Amount of Money Bet in Testing

Figure 5.87 shows the frequency of different amounts of money bet in the testing process. Similarly, the agent tends to select a lower amount of money to bet as shown in figure 5.76 while partial of the frequency of selecting $20 is changed to selecting $30.

Win Rate

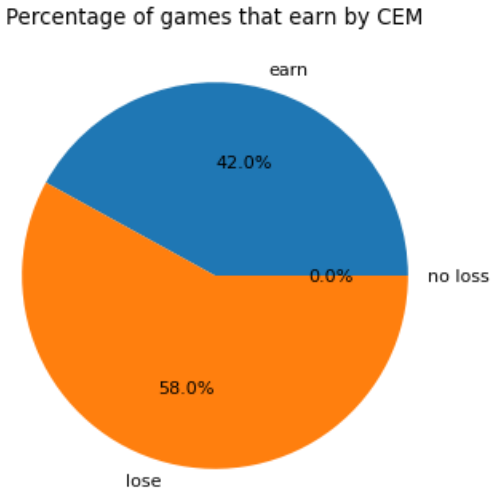

Figure 5.88 Win Rate of Agent in Training

Figure 5.88 shows the percentage of winning, losing and not betting in the training process. The agent achieves a high winning rate of 42.0% which is lower than the result of DQN by 0.1% in the same environment.

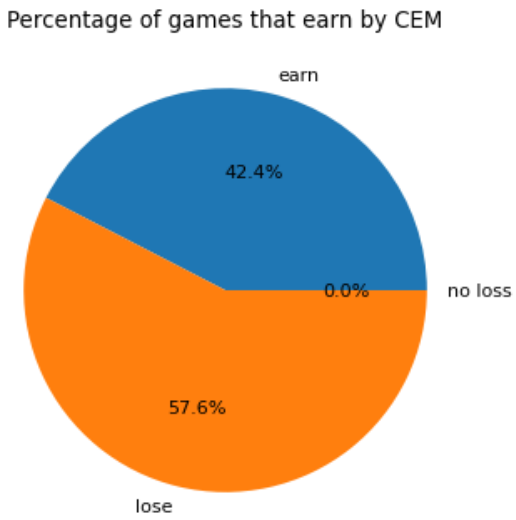

Figure 5.89 Win Rate of Agent in Testing

Figure 5.89 shows the percentage of winning, losing and not betting in the testing process which has a slightly better performance than the training process. The winning rate rose by 0.4% which explained the better profitability in the testing phase.

5.2.5.4 Summary

In summary, although the cumulative reward of Cross Entropy Method in both environments are not converging after all timesteps trained while the overall performance of CEM agent is still comparable to other agents, especially for the profitability of CEM agent in type 2 environment.

On the other hand, the horse betting strategies of CEM agents in both environments are quite different since the agent selected more diversified actions with various levels of risks in the type 1 environment while the agent selects horse 2 mostly which is a safer strategy. This also reflects that the CEM agent is able to obtain different betting strategies in different environments based on the rewards and penalties. Besides, the amount of money selected by the agent also shows that the CEM agent tends to play a defensive betting strategy in type 2 environment and gets an outstanding result compared to other agents.

5.3 Comparison of results among all algorithms

In this chapter, we would compare the algorithms side by side with the same environment settings and attempt to evaluate their strength from different aspects.

The environment setting to be adopted would be Environment (Type 1) in 5.2.1.1 where only 1 horse is allowed to be bet and amount of money to be bet is fixed. No advance settings are chosen for two reasons: 1. It wasn't successful using one bandit algorithm to bet a flexible amount of money without tricky constructs. 2. Analyzing the results of choosing more than 1 horse is more difficult.

5.3.1 Comparison

In this chapter, we will compare the best 2 out of all bandit algorithms which are neural epsilon and neural UCB, and RL algorithms.

5.3.1.1 Metrics

Besides regrets and rewards that are the main focus of analysis in previous part, we also include counts for optimal/sub-optimal/not-optimal actions played by each algorithm for direct performance comparison. Optimal actions are defined as betting on the horse that gets into the top 3 places of the race and has the highest return reward. Sub-optimal actions are defined as betting on the horse that gets into the top 3 places of the race. Non-optimal actions are defined as betting on the horse that are not the top 3 placed horses or not betting options.

Apart from that, we would also want to evaluate how similar the behaviors of algorithms are. To do that, we use Pearson correlation coefficient on their histories of played actions. Pearson correlation measures the similarity of the trends of two variables.

$$\rho_{X,Y} = \operatorname{corr}(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \tag{16}$$

5.3.1.2 Results and Analysis

Cash Balance

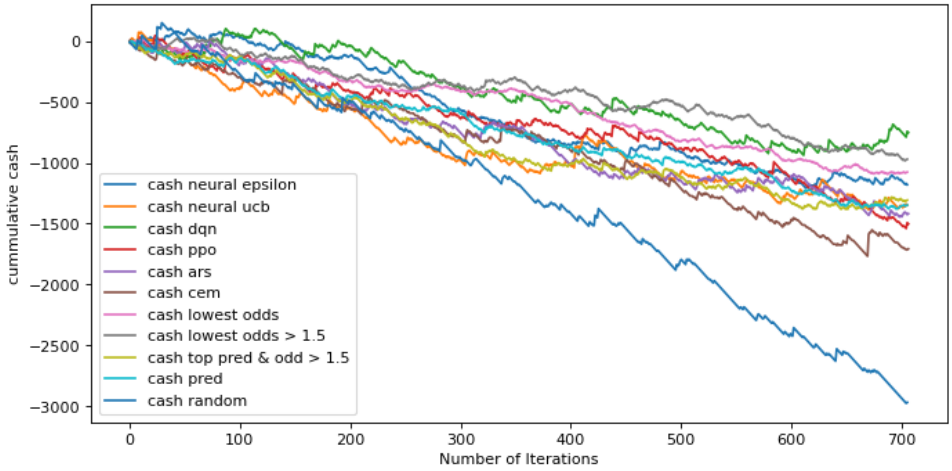

Figure 5.90 Comparison of cumulative cash

None of the algorithms manage to have positive gain. Among all algorithms, DQN performs the best while CEM performs the worst. For other algorithms, the performances of them are very similar while the final profits are slightly lower than betting on lowest odds. And interestingly, simply by betting on lowest odds when the odd is larger than the threshold of 1.5 can already surpass most algorithms.

Optimal / Sub-optimal / Non-optimal Actions Counts

Table 5.1 Optimal / Sub-optimal / Non-optimal Actions Counts

	Optimal	**Sub-optimal**	**Non-optimal**
Neural Epsilon	53 (7.50%)	243 (34.37%)	411 (58.13%)
Neural UCB	65 (9.19%)	117 (16.55%)	525 (74.26%)
DQN	59 (8.35%)	205 (29.00%)	443 (62.66%)
PPO	63 (8.91%)	197 (27.86%)	447 (63.22%)
ARS	89 (12.59%)	127 (17.96%)	491 (69.45%)
CEM	59 (8.35%)	84 (11.88%)	564 (79.77%)

Overall, DQN and PPO has highest optimal and sub-optimal total count.

For optimal count, ARS is significantly higher than others while other algorithms are not able to achieve a 10% of optimal actions counts. For sub-optimal count, neural epsilon performs the best.

Pearson Correlation Coefficient between Action Histories

Table 5.2 Pearson Correlation Coefficient between Action Histories

	Neural Epsilon	**Neural UCB**	**DQN**	**PPO**	**ARS**	**CEM**
Neural Epsilon	1.00	0.00	0.05	-0.04	/	-0.03
Neural UCB	0.00	1.00	-0.10	-0.08	/	0.03
DQN	0.05	-0.10	1.00	0.06	/	0.12
PPO	-0.04	-0.08	0.06	1.00	/	0.07
ARS	/	/	/	/	/	/
CEM	-0.03	0.03	0.12	0.07	/	1.00

Pearson Correlation Coefficient between Optimality of Action Histories

Table 5.3 Pearson Correlation Coefficient between Optimality of Action Histories

	Neural Epsilon	**Neural UCB**	**DQN**	**PPO**	**ARS**	**CEM**
Neural Epsilon	1.00	-0.01	0.05	-0.04	-0.01	-0.05
Neural UCB	-0.01	1.00	0.02	0.03	0.04	-0.06
DQN	0.05	0.02	1.00	0.08	-0.05	-0.00
PPO	-0.04	0.03	0.08	1.00	-0.08	-0.08
ARS	-0.10	0.04	-0.05	-0.08	1.00	0.44
CEM	-0.05	-0.06	-0.00	-0.08	0.44	1.00

For Pearson correlation coefficient, there is no obvious observations for both action histories and optimality of action histories except that ARS and CEM has high Pearson correlation coefficient on optimality, which might suggests their similarity in tendency of choosing top 3.

5.3.2 Model Selection using Bandit Algorithm

The performance of reinforcement learning algorithms is not always stable as the environment is changing. At different intervals, the best performing algorithm might be different. Therefore, ideally, we would hope to always select actions from the best algorithm at any time point instead of hanging on a single option. In order to do so, we would again take advantage of the bandit algorithm. More specifically, we would use EXP3 as stated in chapter 2.

5.3.2.1 Experiment Settings

The data to be used is the action histories of all the algorithms on the test dataset with 707 races. In each round, EXP3 selects one algorithm and plays the action played by the algorithm. The reward received is the same as the reward received by playing the action.

5.3.2.2 Results

Regret Choosing Algorithm

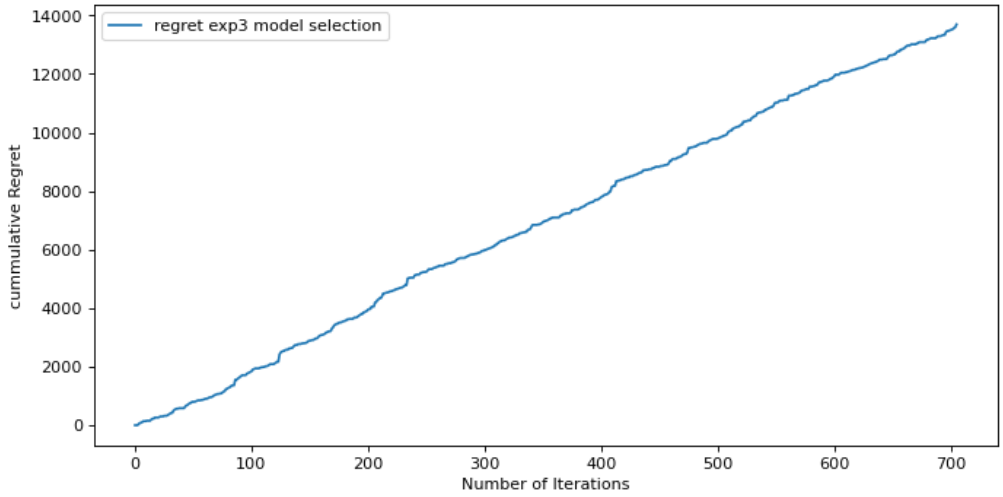

Figure 5.91 Cumulative regret of using EXP3

Cash Balance

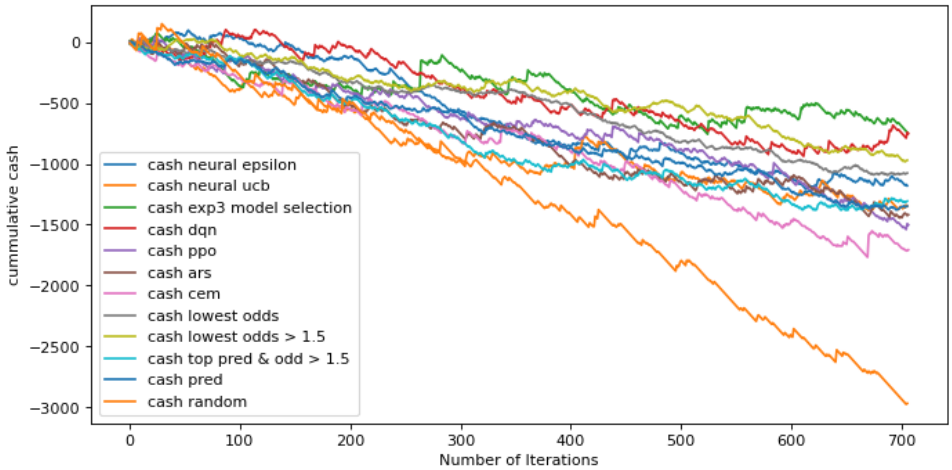

Figure 5.92 Comparison of cumulative cash with EXP3

Performance of applying EXP3 for model selection is better than that of any single algorithm.

Counts of Actions taken by EXP3 and Trends over Time

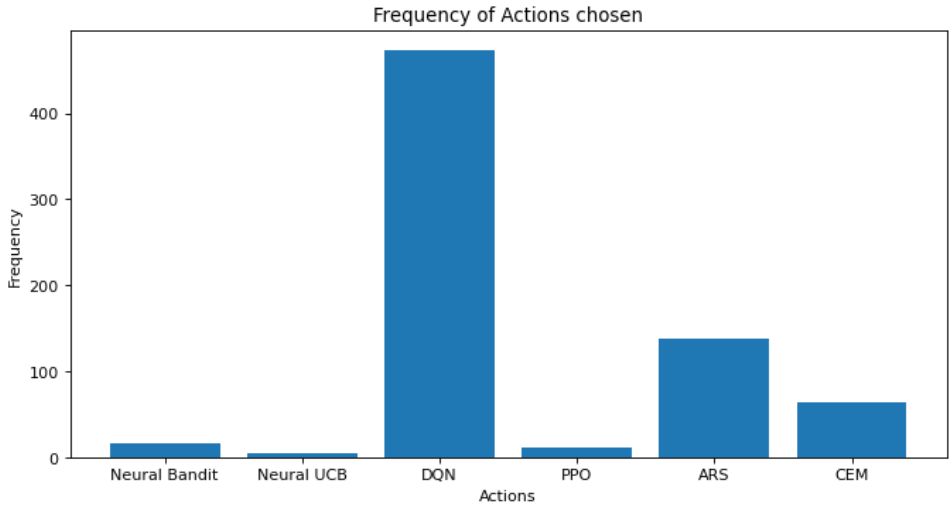

Figure 5.93 Actions counts when using EXP3

Among all selections, DQN is chosen the most by EXP3. These matches the acceptable win rate of DQN in testing phase shown in section 5.2.2.2 which is 36.4% and the final profits show in previous graphs.

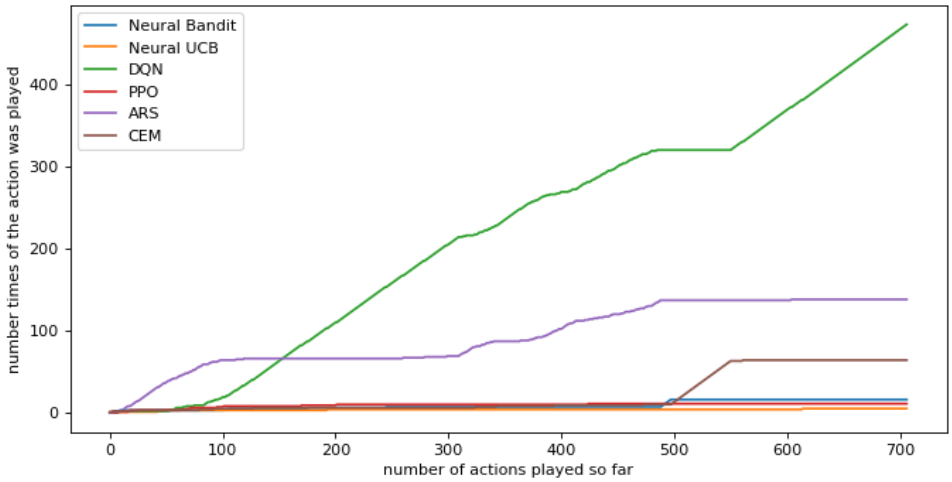

Figure 5.94 Trend of actions selection

ARS is chosen the most at first and later DQN is chosen significantly more. And for others, they are barely chosen.

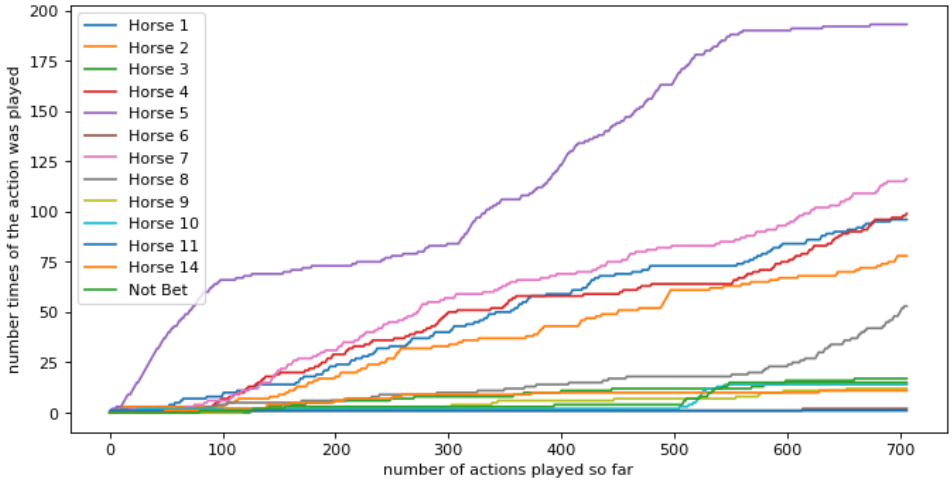

Figure 5.95 Trend of horse selection

Among all, horse 5 is chosen the most. And followed by 7, 11, 4 and 2. Most of these are fewer safe options but yet EXP3 doesn't lose too much in the end.

Optimal / Sub-optimal / Non-optimal Actions Counts

Table 5.4 Optimal / Sub-optimal / Non-optimal Actions Counts

	Optimal	**Sub-optimal**	**Non-optimal**
Neural Epsilon	53 (7.50%)	243 (34.37%)	411 (58.13%)
Neural UCB	65 (9.19%)	117 (16.55%)	525 (74.26%)
EXP3 Model Selection	59 (8.35%)	179 (25.32%)	469 (66.34%)
DQN	59 (8.35%)	205 (29.00%)	443 (62.66%)
PPO	63 (8.91%)	197 (27.86%)	447 (63.22%)
ARS	89 (12.59%)	127 (17.96%)	491 (69.45%)
CEM	59 (8.35%)	84 (11.88%)	564 (79.77%)

Despite good results of EXP3, we don't see the number of optimal or sub-optimal actions of EXP3 be better than others. This might be because of the quality of actions it chooses is higher.

Pearson Correlation Coefficient between Action Histories

Table 5.5 Pearson Correlation Coefficient between Action Histories

	Neural Epsilon	**Neural UCB**	**EXP3 Model Selection**	**DQN**	**PPO**	**ARS**	**CEM**
Neural Epsilon	1.00	0.00	0.00	0.05	-0.04	/	-0.03
Neural UCB	0.00	1.00	-0.06	-0.10	-0.08	/	0.03
EXP3 Model Selection	0.00	-0.06	1.00	0.66	0.08	/	0.24
DQN	0.05	-0.10	0.66	1.00	0.06	/	0.12
PPO	-0.04	-0.08	0.08	0.06	1.00	/	0.07
ARS	/	/	/	/	/	/	/
CEM	-0.03	0.03	0.24	0.12	0.07	/	1.00

Pearson Correlation Coefficient between Optimality of Action Histories

Table 5.6 Pearson Correlation Coefficient between Optimality of Action Histories

	Neural Epsilon	**Neural UCB**	**EXP3 Model Selection**	**DQN**	**PPO**	**ARS**	**CEM**
Neural Epsilon	1.00	-0.01	0.02	0.05	-0.04	-0.01	-0.05
Neural UCB	-0.01	1.00	0.01	0.02	0.03	0.04	-0.06
EXP3 Model Selection	0.02	0.01	1.00	0.70	0.04	0.15	0.17
DQN	0.05	0.02	0.70	1.00	0.08	-0.05	-0.00
PPO	-0.04	0.03	0.04	0.08	1.00	-0.08	-0.08
ARS	-0.10	0.04	0.15	-0.05	-0.08	1.00	0.44
CEM	-0.05	-0.06	0.17	-0.00	-0.08	0.44	1.00

EXP3 has highest Pearson correlation coefficient with DQN. This matches with previous results and isn’t surprising.

5.3.3 Summary

In summary, exploring horse betting strategies on PLACE using reinforcement learning algorithms and multi-armed bandit algorithms seems to have similar performance in terms of profitability, except DQN performs slightly better than betting on lowest odds. However, all betting strategies are not able to make any profits with PLACE betting. Surprisingly, there is one betting strategy that bets on the horses with the lowest odd, where the odd must be greater than 1.5, has a performance greater than all horse betting strategies except DQN.

By viewing the optimal, sub-optimal and non-optimal actions counts, we can see that although ARS achieves the highest optimal action counts, the profitability of it is has not much difference compared to other algorithms since the return cash of PLACE has little difference between the top 3 placed horse usually. Reversely, CEM achieves the highest non-optimal action count, which has the lowest profit shown in the previous figures.

For the model selection using EXP3, it has been shown that EXP3 successfully learns from the most profitable algorithm DQN among all algorithms which at the end has a cash balance slightly greater than the DQN has. The performance of EXP3 is acceptable since the win rate of other reinforcement learning algorithms are not high enough but EXP3 still can achieve a cash balance that is greater than most of the algorithm for most of the time in the experiment. Thus, if the accuracy of other algorithms can be improved, the profitability of EXP3 can potentially be enhanced.

6. Conclusion and Future

6.1 Conclusion

This report focuses on both horse racing prediction with random forest and exploring betting strategy with multi-armed bandit algorithm and other reinforcement learning algorithms. For horse racing prediction, the accuracy of prediction is acceptable compared to the previous projects and the profitability of betting simulation based on the prediction has been improved in this semester. On the other hand, the interpretability of the random forest has been enhanced, which helps figuring out the features which significantly affect the performance of the horses. Meanwhile, the bandit algorithms are able to find out options that have high expected reward under proper settings of action set and reward function, but we are not able to observe any particularly surprising behaviors from bandit algorithms. For other reinforcement learning algorithms, only comparable performance can be observed throughout the experiments. At last, we attempted model selection using EXP3 to enhance the stability of horse betting strategies which returns an acceptable result.

6.2 Limitations

First of all, since the training and testing datasets are split according based on the race season, and the accuracy of the time prediction model would be impacted by this since the average finishing time of horses decreases by years. As the training dataset includes only the records from 2008 to 2018 and the testing datasets includes only the records from 2019 to 2020. Thus, unseen testing data would be out of the range of the training data which the random forest model is not able to handle, and the accuracy of the prediction would be affected.

In addition, bandit algorithms have strong theoretical guarantees (such as sublinear regret) under constraints (bounded reward, probability distribution of reward is stationary). In our problem, these constraints might not hold. Therefore, we can see that metrics like regret can hardly follow these theoretical expectations. To a certain extent, they still work as what have shown but only in a limited way.

Furthermore, bandit algorithms are good at quickly finding the best average choices, but in horse racing, these options would lose money unless we know better how the horse racing outcome is. Therefore, it’s certainly not enough to earn money by using bandit algorithms at betting. However, it can possibly be combined with other approaches to exploit the results.

On the other hand, overfitting has been a main problem found in the experiments of exploring horse betting strategies using other reinforcement learning algorithms, for instance, Deep Q Network, Proximal Policy Optimization. As stated before, the number of records is limiting and the features of the races are mostly distinct which hinders the learning procedure of agents and returns results that lacks generalization. Thus, the horse betting strategies generated by different algorithms cannot perform as well as expected.

6.3 Problems Encountered

For data collection, the official data provided by the Hong Kong Jockey Club is limited for instance, the age of the horses when they participated in a particular race is not provided. From LYU1805, the age distribution of winning horse has a peak on horses aged 4 and decreases afterward which shows that the age could be an important feature for time prediction [59]. Besides, odds provided by the HKJC included only the final WIN records without fluctuation and PLACE odds. The odds fluctuation data collected from hkHorse is also adequate since only records for previous 2 race seasons are included. Thus, the performance of the models would be affected by the inadequate data.

There are limited existing libraries that have support for variety of bandit algorithms and good flexibility to modify for our own use like combinatorial scenario, defining a horse betting environment, etc. So, we had to spend a lot of time on building the infrastructure from scratch and there is only a handful of algorithms we can try under time constraint. Exploring the bandit algorithms requires good grasp of related theories, the more advanced algorithms are usually new and with no open-source implementations. It’s time consuming and difficult to understand and implement them. Therefore, we only tried more fundamental contextual bandit algorithms this semester.

6.4 Possible Future Directions

Increase accuracy of finishing time prediction

The accuracy of the finishing time prediction from the random forest model is comparable to the previous project while the betting simulation shows the performance can only be comparable to the public intelligence. This performance cannot provide a good environment for exploring horse betting strategies using multi-armed bandits or other reinforcement learning algorithms which is stated in the previous sections.

Different Types of Bets

The betting option we focused on this project is PLACE, which has the lowest profitability among all general betting options provided by the HKJC. In order to gain more profits, different betting options such as Quinella or Trio which would have higher returns.

Inverse Reinforcement Learning

The current approach of our project is to use reinforcement learning to explore the horse betting strategies while the general performances of all algorithms are not outstanding. Instead of learning a decision process, Inverse reinforcement learning, where the agent learns the reward function, could be tried and may return surprising results.

More Data

Currently we just have 1414 races that are with real time odds, which might not be enough. We can grasp more data or find out ways to generate realistic data.

7. Personal Contribution

Throughout the project, we have worked on both horse prediction part and horse betting part. For the finishing time prediction section, I am responsible for doing research for the datasets that are usable for our project and also crawling all the data from the Hong Kong Jockey Club and other data sources mentioned in chapter 3. After that, I have done several basic analyses regarding relationships between finishing time and race classes, odds and also prepared exponential moving average of the odds which are used in both finishing time prediction and horse betting sections. Furthermore, I worked on the random forest model that build by my groupmate and made improvements on it by optimizing input data, parameters. Also, I am responsible for the evaluation and interpretation parts of the random forest model using different metrics, for instance, partial dependence plot, betting simulation etc. Besides finishing time prediction part, I have also worked on exploring horse betting strategies using different reinforcement learning algorithms including implementing and analyzing other than bandit algorithms that my groupmate worked on.

References

[1] A. Fielding and C. A. O'Muircheartaigh. Binary segmentation in survey analysis with particular reference to AID. The Statistician, 25:17–28, 1977.

[2] Agrawal, Shipra, and Navin Goyal. "Thompson sampling for contextual bandits with linear payoffs." International Conference on Machine Learning. PMLR, 2013.

[3] Allesiardo, Robin, Raphaël Féraud, and Djallel Bouneffouf. "A neural networks committee for the contextual bandit problem." International Conference on Neural Information Processing. Springer, Cham, 2014.

[4] Ando Saabas. "Prediction intervals for Random Forests." DataDive, 2 June 2015, https://blog.datadive.net/prediction-intervals-for-random-forests/.

[5] Appel, G.: Technical Analysis: Power Tools for Active Investors. FT Press, Prentice Hall, New York (2005)

[6] Archer, Kellie & Kimes, Ryan. (2008). Empirical characterization of random forest variable importance measures. Computational Statistics & Data Analysis. 52. 2249-2260. 10.1016/j.csda.2007.08.015.

[7] Archontoulis, S.V. and Miguez, F.E. (2015), Nonlinear Regression Models and Applications in Agricultural Research. Agronomy Journal, 107: 786-798. https://doi.org/10.2134/agronj2012.0506

[8] Auer, Peter, et al. "The nonstochastic multiarmed bandit problem." SIAM journal on computing 32.1 (2002): 48-77.

[9] Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem." Machine learning 47.2 (2002): 235-256.

[10] Borowski, Piotr & Chlebus, Marcin. "MACHINE LEARNING IN THE PREDICTION OF FLAT HORSE RACING RESULTS IN POLAND." 2021, 10.13140/RG.2.2.22254.95043.

[11] Breiman, Leo. "Bagging predictors." Machine learning 24.2 (1996): 123-140.

[12] Breiman, Leo. "1 RANDOM FORESTS--RANDOM FEATURES." (1999).

[13] Cheetham, J., et al. "Relationships between race earnings and horse age, sex, gait, track surface and number of race starts for Thoroughbred and Standardbred racehorses in North America." Equine veterinary journal 42.4 (2010): 346-350.

[14] Chen, Lixing, Jie Xu, and Zhuo Lu. "Contextual combinatorial multi-armed bandits with volatile arms and submodular reward." Advances in Neural Information Processing Systems 31 (2018): 3247-3256.

[15] Cheng, T., Lau, M. “Predicting Horse Racing Result Using TensorFlow” 2017

[16] Cheung, K., & Or, K. “Applying Reinforcement Learning to “Play” Horse Racing” (2020).

[17] Couronné, Raphael & Probst, Philipp & Boulesteix, Anne-Laure. (2018). Random forest versus logistic regression: A large-scale benchmark experiment. BMC Bioinformatics. 19. 10.1186/s12859-018-2264-5.

[18] Dan He and L. Parida, "Does encoding matter? A novel view on the quantitative genetic trait prediction problem," 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2015, pp. 123-126, doi: 10.1109/BIBM.2015.7359667.

[19] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, Nitin Motgi, Seung-Taek Park, Raghu Ramakrishnan, Scott Roy, and Joe Zachariah. Online models for content optimization. In Advances in Neural Information Processing Systems 21 (NIPS-08), pages 17–24, 2009

[20] Elroy Dimson, Massoud Mussavian, "MARKET EFFICIENCY," in THE CURRENT STATE OF BUSINESS DISCIPLINES, SPELLBOUND PUBLICATIONS, 2000, pp.959 – 970

[21] Fatima, M. and Pasha, M. “Survey of Machine Learning Algorithms for Disease Diagnostic.” Journal of Intelligent Learning Systems and Applications, 9, 1-16., 2017, https://doi.org/10.4236/jilsa.2017.91001

[22] Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., & Dabney, W. (2020, November). Revisiting fundamentals of experience replay. In International Conference on Machine Learning (pp. 3061-3071). PMLR.

[23] Friedman, Jerome H. “Greedy function approximation: A gradient boosting machine.” Annals of statistics (2001): 1189-1232.

[24] Giovanni Seni; John Elder, Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions , Morgan & Claypool, 2010.

[25] Greenwell, Brandon M., Bradley C. Boehmke, and Andrew J. McCarthy. “A simple and effective model-based variable importance measure.” arXiv preprint arXiv:1805.04755 (2018).

[26] Hackeling, Gavin. Mastering Machine Learning with scikit-learn. Packt Publishing Ltd, 2017.

[27] Ho, Tin Kam. "Random decision forests." Proceedings of 3rd international conference on document analysis and recognition. Vol. 1. IEEE, 1995.

[28] Hoffman, Matthew, Eric Brochu, and Nando de Freitas. "Portfolio Allocation for Bayesian Optimization." UAI. 2011.

[29] Hong Kong Jockey Club. "Course Information - Reference Information - Horse Racing." https://racing.hkjc.com/racing/english/racing-info/racing_course.aspx

[30] Hong Kong Jockey Club. "Equipment Register (Equine and Human)" https://racing.hkjc.com/racing/english/racing-info/rules_pdf/Equipment-Register-Full-2021-English.pdf. 09/2021

[31] Hong Kong Jockey Club. "Handicapping Policy - Reference Information - Horse Racing - The Hong Kong Jockey Club." https://racing.hkjc.com/racing/english/racing-info/handicap_policy.asp

[32] Hong Kong Jockey Club. "Racing Forward Together", https://corporate.hkjc.com/corporate/common/annualreport/pdf/HKJC_AR2021_Full.pdf. 2021

[33] István Szita, András Lörincz; Learning Tetris Using the Noisy Cross-Entropy Method. Neural Comput 2006; 18 (12): 2936–2941. doi: https://doi.org/10.1162/neco.2006.18.12.2936

[34] Khalid, Samina, Tehmina Khalil, and Shamila Nasreen. "A survey of feature selection and feature extraction techniques in machine learning." 2014 science and information conference. IEEE, 2014.

[35] Laurae. "Visiting: Categorical Features and Encoding in Decision Trees." Medium, 20 June 2018, medium.com/data-design/visiting-categorical-features-and-encoding-in-decision-trees-53400fa65931.

[36]. Li, Lihong, et al. "A Contextual-Bandit Approach to Personalized News Article Recommendation." Proceedings of the 19th International Conference on World Wide Web - WWW '10, 2010, https://doi.org/10.1145/1772690.1772758.

[37] Liu, Y., Wang, Z. "Predicting Horse Racing Result with Machine Learning" 2017

[38] Livingstone, D., Manallack, D. & Tetko, I. Data modelling with neural networks: Advantages and limitations. J Comput Aided Mol Des 11, 135–142 (1997). https://doi.org/10.1023/A:1008074223811

[39] Loh, Wei-Yin. "Classification and regression trees." Wiley interdisciplinary reviews: data mining and knowledge discovery 1.1 (2011): 14-23.

[40] Luengo, Julián, Salvador García, and Francisco Herrera. "A study on the use of imputation methods for experimentation with radial basis function network classifiers handling missing

attribute values: The good synergy between rbfns and eventcovering method." Neural Networks 23.3 (2010): 406-418.

[41] Majumdar, J., Naraseeyappa, S., et al. “Analysis of agriculture data using data mining techniques: application of big data.” J Big Data 4, 20, 2017, https://doi.org/10.1186/s40537-017-0077-4

[42] Mania, H., Guy, A., & Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning. arXiv preprint arXiv:1803.07055.

[43] Marmerola, Guilherme Duarte. “Introduction to Thompson Sampling: The Bernoulli Bandit.” Guilherme's Blog, 21 Nov. 2017, https://gdmarmerola.github.io/ts-for-bernoulli-bandit/.

[44] Meinshausen, N., & Ridgeway, G. (2006). Quantile regression forests. Journal of Machine Learning Research, 7(6).

[45] Motulsky, H.J. and Ransnas, L.A. (1987), Fitting curves to data using nonlinear regression: a practical and nonmathematical review. The FASEB Journal, 1: 365-374. https://doi.org/10.1096/fasebj.1.5.3315805

[46] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

[47] N. M. Allinson and D. Merritt, "Successful prediction of horse racing results using a neural network," IEE Colloquium on Neural Networks: Design Techniques and Tools, 1991, pp. 4/1-4/7.

[48] Naoki Abe, Alan W. Biermann, and Philip M. Long. Reinforcement learning with immediate rewards and linear hypotheses. Algorithmica, 37(4):263–293, 2003.

[49] Nicodemus, K.K., Malley, J.D., Strobl, C. et al. The behaviour of random forest permutation-based variable importance measures under predictor correlation. BMC Bioinformatics 11, 110 (2010). https://doi.org/10.1186/1471-2105-11-110

[50] R. Messenger and L. Mandell. A modal search technique for predictive nominal scale multivariate analysis. Journal of the American Statistical Association, 67: 768–772, 1972.

[51] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[52] Seger, Cedric. An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing. 2018

[53] Somasundaram, R. S., and R. Nedunchezhian. "Missing value imputation using refined mean substitution." International Journal of Computer Science Issues (IJCSI) 9.4 (2012): 306.

[54] Strobl, C., Boulesteix, AL., Zeileis, A. et al. Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics 8, 25 (2007). https://doi.org/10.1186/1471-2105-8-25

[55] Thaler, Richard H., and William T. Ziemba. "Anomalies: Parimutuel betting markets: Racetracks and lotteries." Journal of Economic perspectives 2.2 (1988): 161-174.

[56] The Hong Kong Jockey Club. “Pari-Mutuel Local Pools - Beginners guide - Betting Entertainment - The Hong Kong Jockey Club”, https://special.hkjc.com/racing/info/en/betting/guide_qualifications_pari.asp

[57] The Hong Kong Jockey Club. “Racing Novice Class Racing 101”, https://entertainment.hkjc.com/entertainment/common/chinese/images/more-about-racing/racing-101/Racing-101_201509.pdf. 2015

[58] W. Chung, C. Chang and C. Ko, "A SVM-Based committee machine for prediction of Hong Kong horse racing," 2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media), 2017, pp. 1-4, doi: 10.1109/UMEDIA.2017.8074091.

[59] Wong, Y. “Horse Racing Prediction using Deep Probabilistic Programming with Python and PyTorch”, 2018

[60] Zhao, Qingyuan, and Trevor Hastie. “Causal interpretations of black-box models.” Journal of Business & Economic Statistics, to appear. (2017).

[61] Zheng, Alice, and Amanda Casari. Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. 1st ed., O’Reilly Media, 2018.

[62] Zhou, Dongruo, Lihong Li, and Quanquan Gu. "Neural contextual bandits with ucb-based exploration." International Conference on Machine Learning. PMLR, 2020.

[63] “3.3. Metrics and Scoring: Quantifying the Quality of Predictions.” Scikit-Learn, scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error. Accessed 30 Nov. 2021.

[64] “Feature Importances with a Forest of Trees.” Scikit-Learn, 2007, scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html.

[65] “Hong Kong Jockey Club.” Wikipedia, Wikimedia Foundation, 06/10/2021, https://en.wikipedia.org/wiki/Hong_Kong_Jockey_Club