The Chinese University of Hong Kong

Final Year Project Report (Term 2)

Betting Odds Calculation with Machine Learning

Author:

NAM Man Leung

Supervisor:

Prof. LYU Rung Tsong Michael

LYU2102

Faculty of Engineering

Department of Computer Science and Engineering

20 April 2022

The Chinese University of Hong Kong

Abstract

Faculty of Engineering

Department of Computer Science and Engineering

BSc degree in Computer Science

Betting Odds Calculation with Machine Learning

by Nam Man Leung

Win odds in horse racing reflect public opinions because the more confidence the public has about winning a horse, the lower the win odds due to the pari-mutuel betting system. The transformer model in the natural language process has successfully dealt with the prediction of sequence input. Still, there is no research exploring the transformer model's use in horse racing prediction. The ratings given by rating systems have been used in many competitions to represent the skill level of players. In this project, we combine these two techniques for horse racing prediction and see if it can have the same effect as the win odds in helping the forecast. By comparing the results of using each technique alone, we show that combining the two approaches can achieve better prediction accuracy and positive net gain in betting simulation. Furthermore, horse token embedding is proposed to replace the word embedding in the transformer model to boost the prediction accuracy and maintain a steady growth of net gain in betting simulation. To develop a deeper understanding of the transformer model, probing, analysis of attention map, and integrated gradient are methods used to assess model capability, data shuffling impact, and input-output behaviors.

Acknowledgment

I want to express my gratitude to my supervisor, Professor Michael R. LYU, and my advisor, Mr. HUANG, who provided suggestions and guidance throughout the project.

I would also like to show appreciation for previous students' horse racing prediction work, which strengthens my background knowledge in doing this project.

Table of Contents

Chapter 1

Overview

Reproducing the effects of win odds in horse racing prediction with machine learning methods and understanding the underlying mechanism and behaviors of those methods are the purposes of this final year project. As the win odds keep changing before the start of a horse race, data collection of the win odds may not be accurate enough before the race. However, betting is not permitted after the beginning of the race. Hence, we attempt to use only static data which do not vary within the betting period, combined with a neural network to resemble the helpfulness of win odds in horse racing prediction in the first stage. In the next stage, we endeavor to understand the underlying mechanism and behaviors of the neural network used for horse racing prediction. The introduction to machine learning methods and the background about horse racing are provided at the beginning of this section. Then, our motivation for this project and the respective objectives of the first and second semesters will be stated.

1.1 Introduction

Machine learning has become a hot topic in technical fields with the dramatic advancement of the hardware and appearance of big data in recent years. It has been applied to real-world problems such as weather forecast, image recognition, speech recognition, natural language process, etc. The concept of machine learning is optimizing the parameters defined in a model with the guidance of training experience to get intuition and prediction [1]. Machine learning is not specific to one particular

field but the junctions of different domains such as statistics, computer science, and data science. For instance, it uses the knowledge in statistics to build models and knowledge in computer science to convert the models into computer representations and design an efficient algorithm to deal with the optimization problem of the model [2].

Machine learning can be divided into two types. The first type is supervised learning, in which the known target outputs are used to correct the values of the parameters in the mapping model between the input and outcome [3]. The mapping will then be employed for predicting the output of new incoming data. The second type is unsupervised learning, in which there is no explicit target output to guide the optimization of parameters in the model. Instead, an assessment of the representation's quality is learned in a self-organizing process [4]. Supervised learning is our choice for this project because the win odds can easily be collected from the HKJC website.

The primitive neural network architecture in machine learning was the single-layer perceptron proposed in 1958. It was further developed into a multilayer perceptron in 1975 to solve nonlinear problems and linearly sparable problems that the perceptron cannot solve [5]. The multilayer perceptron gradually evolves to different neural network architecture such as deep neural network, convolutional neural network, recurrent neural network, and long/short term memory network. The original design of the neural network was to emulate how the brain function in doing a task by treating each neuron in the neural network as the neuron in the brain and aggregating them into a complicated information system that is nonlinear [6].

Natural Language processing has been a popular topic in the research field. It had initially addressed by the convolutional neural network and recurrent network due to their exceptional performances until the appearance of the transformer architecture in 2017, which has an even better performance in understanding and generating the natural language by parallel training and the ability to tackle lengthy sequence inputs [7].

Previous FYP students made several attempts in horse racing prediction with machine learning methods. LYU1603 tried to predict the winning horse with regression on time [8]. LYU1703 attempted to predict the winning horse and the places with MLP and rank network [9]. LYU1805 tried to predict the winning horse with deep probabilistic programming [10]. We approach the horse racing prediction from a different perspective for this project. Since both the inputs of this horse racing prediction and natural language processing are sequences, we decide to reduce the horse racing prediction to a natural language processing classification problem. We hope that the techniques in natural language processing can capture the relationships between horses in a single race and make the prediction according to the dependency.

This project makes four contributions. The first contribution is applying the transformer model in horse racing prediction, which has not been explored yet. The second contribution proposes a new embedding method suitable for horse racing data. The second contribution shows a positive net gain when using the prediction of the transformer model with ratings as input in horse racing betting. The fourth contribution is revealing the behaviors of the transformer model with horse racing data via different interpretability methods.

1.2 Background of Horse Racing

1.2.1 Horse Racing in Hong Kong

Horse racing in Hong Kong is a sports competition introduced by the British, which usually has 10 – 14 jockeys riding on corresponding horses in a single race, competing to reach the finish line in a shorter time. It has been an esteemed sports event in Hong Kong for over 100 years as betting allows people to bet on the horses they like. This event is mainly held on Sundays and Wednesdays. There are 10-day races on Sundays and 8-night races on Wednesdays, respectively. The number of competitors is limited to 14 for races on Sundays, while it is limited to 12 for races on Wednesdays. Each year, the horse racing season starts in September and ends in July, and roughly 88 days have the horse racing within a season [11].

1.2.2 The Hong Kong Jockey Club

The Hong Kong Jockey Club, founded in 1884, is a certified non-profit making and charitable organization responsible for hosting horse racing events and other betting entertainments. It gains enormous revenue from its sport betting events every year, and those revenues will be split for operational costs and returned to the community. HK$29.4 billion was returned to the community regarding duty, tax, and donations in 2020-2021 [12].

1.2.3 Pari-mutuel Betting in Horse Racing.

In pari-mutuel betting, the bets from people are accumulated in a pool in each race. The bookmaker will take a fixed percentage from the pool [13]. In Hon Kong, The Hong Kong Jockey Club acquires 17.5% of the pool in winning bets as its revenue and allocates the remaining in the pool to the betters with a correct prediction concerning the odd, which is the ratio of return to the bet calculated before the start of the race. The odds cannot be interpreted as the actual winning probability of a horse, but it is just an estimation of how many betters favors the horse. In other words, it reflects public intelligence. Since the betters are betting against each other, a positive net gain is expected if we make a more accurate prediction than the public [14].

1.2.4 Types of Bets

The Hong Kong Jockey Club provides various types of bets for bettors. The types and explanations can be found in Figure 1.

		Telephone Betting	Interactive Services#	Off-Course Betting Branches		Racecourses	
				Self Vending Terminal	Service Counter	Self Vending Terminal	Service Counter
Minimum Investment Amount	Horse Racing	Pari-Mutuel Pools					
		From 30 minutes before Race 1 up to the last race starts, • HK$20 until 10 minutes before the start of each race • HK$50 during the last 10 minutes of each race	Applicable IS Device(s) HK$10(any time)	HK$10 (any time)	HK$20 (30 mins before Race 1 to start of last race)	HK$10 (any time)	HK$10 (any time)

Figure 1. Pari-mutuel betting provided by the HKJC [15]

As we see from Figure 1, the minimum amount to invest in the pari-mutuel pools is $10 from a self-vending terminal such as the HKJC mobile application or the HKJC WEB Application at any time.

Single-race Pool	Dividend Qualification
Win	1st in a race
Place	1st, 2nd or 3rd in a race, or 1st or 2nd in a race of 4 to 6 declared starters (applicable to local races) 1st, 2nd, 3rd or 4th in a race, or 1st, 2nd or 3rd in a race of 7 to 20 declared starters, or 1st or 2nd in a race of 4 to 6 declared starters (applicable to designated simulcast races)
Quinella	1st and 2nd in any order in a race
Quinella Place	Any two of the first three placed horses in any order in a race
3 Pick 1 (Composite Win) **Winning Trainer (Composite Win)** **Winning Region (Composite Win)**	Composite containing the 1st horse in a race
Forecast	1st and 2nd in correct order in a race
Trio	1st, 2nd and 3rd in any order in a race
Tierce	1st , 2nd and 3rd in correct order in a race
First 4	1st, 2nd , 3rd and 4th in any order in a race
Quartet	1st, 2nd , 3rd and 4th in correct order in a race

Table 1. Types of bets in the single race pool [16]

The single race pool and the dividend qualification for beginners are shown in Table 1.

Multi-race Pool	Dividend Qualification
Double	1st in each of the two nominated races Consolation :1st in 1st nominated race and 2nd in 2nd nominated race
Treble	1st in each of the three nominated races Consolation : 1st in the first two Legs and 2nd in 3rd Leg of the three nominated races

Table 2. Type of bets in multi-race pool [16]

The multi-race pool and the dividend qualification for more experienced bettors are shown in Table 2

1.3 Motivation

Horse racing held by the Hong Kong Jockey Club has been the most favored sports betting event in Hong Kong. Its popularity can be shown to be the colossal amount of revenue, which is approximately HK$280 billion in 2020-2021, despite the economic downturn caused by the coronavirus pandemic [17].

Tremendous efforts have been made to predict the winning horse of each race by machine learning. Still, the outcome has been unsatisfactory as profitable results can only be attained under certain circumstances. It is believed that the betting odds hide the secret of beneficial plans from the observation that bookmakers consistently have interests in providing profitable betting odds to gamblers. Therefore, building and interpreting the machine learning model, which has a similar effect on the betting odds in horse racing prediction, may help reveal the hidden message of the betting odds and the reasons for the bookmaker's enormous financial gain.

1.4 Objectives

We have two objectives in this project. The first objective of this project is to reproduce the effect of win odds from the Hong Kong Jockey Club in horse racing prediction. As the horse with a low winning odd usually has a higher winning probability, as implied from the public intelligence, the win odds contain helpful information which guides the model prediction. However, the dynamic nature of the win odds before the start of a race cannot guarantee the data which we get at a particular time is accurate, and we, therefore, exclude the win odds from our input data and try to reproduce the effect of win odds with only static variables.

The second objective is comprehending the decision made by the model by different interpretability methods. Even though machine learning model usually provides more accurate predictions than humans, we are advised to understand the decision principles and vulnerabilities of the neural network so that we are convinced to apply the model.

First Term:

- Convert the data collected from the HKJC into a sequence that can be fitted to a natural language processing model.
- Find other features that have a similar meaning as the win odds
- Build a natural language process model for winning horse classification
- Evaluate the performance of the proposed model on the test set.

Second Term:

- Improve the winning horse classification by designing a more appropriate embedding scheme for the input
- Inspect the capabilities of the transformer model in the horse racing context
- Examine the vulnerabilities of the transformer model and find the possible underlying reasons
- Enhance the interpretability of the transformer model by extracting intuitive rules from the model decisions.

1.5 Related Works

Researchers have been interested in applying machine learning methods to learn the complex relationship between sports betting and predicting the outcome accurately. Several studies investigated horse racing prediction by the artificial neural network [18], conditional logistic regression [14], random forest [20], and support vector machine [21].

Elnaz and Khanteymoori [18] applied an artificial neural network in horse racing prediction with five supervised neural network learning algorithms: Conjugate Gradient Descent, Quasi-Newton Levenberg-Marquardt, Backward-Propagation, and Backward-Propagation with Momentum. The experiment used the horse racing records in January 2010 in the United States, and the result was exceptional that all learning algorithms produced satisfying predictions of 77% accuracy on average. The performance differences between the learning algorithms are minor. Although Backward-Propagation took a longer training time, it achieved a slightly better prediction result than others. This research demonstrated that artificial neural networks are applied to horse racing prediction.

Silverman and Suchard [14] proposed adjustments to the multinomial logit model for horse racing prediction suggested by Bolton and Chapman [19]. They exploited the winning dividends by introducing a frailty contribution and parameter regularization to the regression model. They collected the data of 3681 races in Hong Kong from the HKJC, and 737 races were retained for testing the model. They discovered that they could gain a remarkable higher return by changing the objective to simply increasing the profit and combining a calculated inverse-frailty score in the experiment.

Lessmann, Sung, and Johnson [20] explored alternative methods for predicting horse racing results. They admitted that the conditional logit model was a proper tool for estimating the winning probability of a horse in conjunction with other horses in a race. In addition to that, they showed that random forest could complement conditional logit-based horseracing forecasting. Consequently, they adapted a two-stage modeling framework that captured the complicated relationship between horses' information and the results of races in the first stage. Then, the winning probability of a horse within a single race was computed at the second stage. In the second stage, a random forest revealed the winner horse by counting the number of votes regarding whether the horse was a winner from the decorrelated decision trees.

Chung, Change, and Ko [21] utilized the support vector machine to predict horse racing results in Hong Kong. They divided their training data into multiple similar training sets and trained a support vector machine for each training set. They were combined to form a more robust model for those weaker models. The outcome of a race was determined similarly to a random forest. All trained support vector machines created a committee machine and did voting. In the experiment, they collected data from the HKJC official website. There were 33532 horse records and 2691 race records dated from 1st Jan 2012 to 30th June 2015 in the dataset. The result of the experiment showed a 70.86% accuracy in predicting the winner horse by the committee machine.

Tung and Hei [8] attempted to build a classification model for winning horse prediction with Tensorflow. They used the neural network to create a binary classification model and betted on the horse if the model's prediction revealed that the horse was a winner. They set a confident threshold to be 0.8 so that they only gambled

the horse when the model predicted it as a winner with a certain threshold exceeding 0.8. As a result, they exhibited a 30% net gain after one year.

Liu [9] tackled the horse racing prediction problem by building a supervised neural network in predicting the finishing time of each horse. After that, he compared horses' anticipated finishing times and ranked them based on their anticipation. He set a confident threshold of 0.5 and betted only on class 1 and 2 races. This setting was shown to have a positive net gain over an entire race season.

Wong [10] applied Pyro, a probabilistic programming language supported by Python, to build sophisticated probabilistic models. The PyTorch backend assisted automatic differentiation, neural networks, and backward propagation. The abstraction provided by the probabilistic programming language simplifies the code for inferences and probabilistic sampling. The result of the experiment showed a profit of 14.43% could be gained when using features including the win odds, while it dropped to 7.59% when using features excluding the win odds.

Chapter 2

Background Knowledge

2.1 Rating Systems

2.1.1 Glicko Rating System

Glicko rating system [22] extends the Elo rating system. It is a statistical model that addresses the limitation of the Elo rating system by introducing an additional measurement of the rating deviation. This measurement is intended for assessing the reliability of a player's rating. When the value of rating deviation is high, it infers that the player has not played the game for an extended period, and the rating thus becomes unreliable. In contrast, a low value of rating deviation indicates that the player plays the game frequently, and the rating is more reliable. The intuition is that the uncertainty of a player's ability reduces because more information is obtained by the player playing more games.

The rating and the rating deviation of horses are calculated in two steps. The formula is recursive as the current rating and rating deviation are determined from the rating and deviation from the last rating and last rating deviation.

During the new rating period, we should compute each horse's rating and rating deviation based on its previous rating and rating deviation. In step 1, we focus on the rating deviation.

If the horse is new to the race, which means it hasn't participated in any races, we

assign 1500 and 350 to its rating and rating deviation respectively. Both 1500 and 350 are default values of the rating and the rating deviation.

If the horse participated in races in the past, we take its rating from the last race for computing the current rating deviation with the formula below,

$$\sigma = min(\sqrt{{\sigma_{old}}^2 + c^2}\,, 350). \quad (1)$$

σ is the current rating deviation and σ_{old} is the rating deviation of the last race.
c is the constant controlling the uncertainty between races. The current rating deviation is the minimum value between the computation from the old rating deviation and 350.

In step 2, we update the rating and rating deviation for each horse in a race.
Let r be the horse's rating in the last race and σ be the rating deviation computed in step 1. Then, $r_1, r_2, \ldots, r_n$ are the rating of the other horses from their last rating period. The corresponding rating deviation is $\sigma_1, \sigma_2, \ldots, \sigma_n$. The result of horses in the race is $s_1, s_2, \ldots, s_n$. If the horse wins the race, s_i equals to one. If the horse loses the race, s_i equals to zero.

Let $r_{new}\ and\ \sigma_{new}$ be the updated rating and rating deviation of a horse and we repeat this procedure for each horse.
We first define the following terms,

$$q = \frac{\ln(10)}{400}\,, \quad (2)$$

$$g(\sigma) = \frac{1}{\sqrt{1+\frac{3q^2(\sigma^2)}{\pi^2}}} \quad , \tag{3}$$

$$E(s|r, r_j, \sigma_j) \; = \frac{1}{1 + 10^{-g(\sigma_j)(r - r_j)/400}} \; , \tag{4}$$

$$d^2 \; = \; (q^2 \sum_{j=1}^{n} (\, g(\sigma_j))^2 \; E(s|r, r_j, \sigma_j) \; (1 - \; E(s|r, r_j, \sigma_j)))^{-1}, \tag{5}$$

The above terms are used in the update.

$$r_{new} \; = \; r \; + \; \frac{q}{1/\sigma^2 + 1/d^2} \sum_{j=1}^{n} g\left(\sigma_j\right) \left(s_j \; - \; E(s|r, r_j, \sigma_j)\right) \; , \tag{6}$$

$$\sigma_{new} \; = \; \sqrt{\left(\frac{1}{\sigma^2} + \; \frac{1}{d^2}\right)^{-1}}. \tag{7}$$

2.1.2 TrueSkill Rating System

TrueSkill rating system [23] also measures the uncertainty of player skill level, but it also has additional features to the Glicko rating system. The first one is the relaxation of the number of players in a game. As the Glicko rating system is designed for 2-players chess games, it assumes that there are only one winner and one loser in each game. The TrueSkill rating system tries to adapt to a multiple-player environment by taking that the outcome of each match is a permutation of multiple teams or players so that it is dedicated to multiplayer games. The second is the inference for individual skills in games requiring players to form groups. Each team only has one player in our situation because horses in horse racing do not constitute a team, and we treat each horse as a team.

We apply the Trueskill rating system in horse racing in which there are n horses { 1, …, n } in a race, and each horse forms a team with only one member.
Let $T := \{T_1, \ldots, T_n\}$ and T_i be the i-th team which has horse i as the only team member so that $T_i \cap T_j = \emptyset$ for $i \neq j$. We also let $R := (r_1, \ldots, r_n)$ be the result of each team in a race. If the i-th horse wins in a race, then $r_i = 1$. Otherwise, $r_i = i$ if the i-th horse gets the i-th place in the race.

As our goal is to estimate the skill level of horses, we would like to calculate the probability that the players have skill level S given the result of the race R and the team assignment T. From the training dataset, we have the race result given the team assignment T and skill level S. Therefore, we can obtain the probability $P(R|$ S, T) of the race with R as the race result and S as the skill level horse all participating horses.

Then, $P(S|\, R, T)$ can be obtained by Bayes' rule,

$$P(S|\, R, T\,) \;=\; \frac{P(R|\, S,T)\; P(S)}{P(R\mid T)}. \qquad (8)$$

We assume the skill level of each horse is a Gaussian distribution with parameters μ_i and σ_i so that $P(S) \;=\; \prod_{i=1}^{n} N(s_i;\, \mu_i, \sigma_i)$. The race performance of each team T_i is actually the race performance of each horse because every team has only one horse as a member. So, the race performance t_i of T_i is modeled as $N(p_i;\, s_i, \beta^2)$. We then order the teams in ascending order based on their rank so that the order of teams is $r_{(1)} \;\leq\; r_{(2)} \;\leq \ldots \leq\; r_{(n)}$. As a result, the probability that the race has outcome R given the team T is the following,

$$\begin{aligned} P(R \mid T) \;&=\; P(R \mid \{T_1, \ldots, T_n\}) \\ &=\; P(\; t_1 \;>\; t_2 \;> \ldots >\; t_n)\,. \end{aligned} \qquad (9)$$

Assume a very simple horse race with 3 teams and each team has only one horse so that $T_1 \;=\; \{1\},\; T_2 \;=\; \{2\}\, and\; \; T_3 \;=\; \{3\}$. Also, team 1 is the winner while team 2 gets the second place and team 3 gets the third place respectively. The joint distribution $P(S, t \mid R\,, T)$ can be represented by the factor graph below.

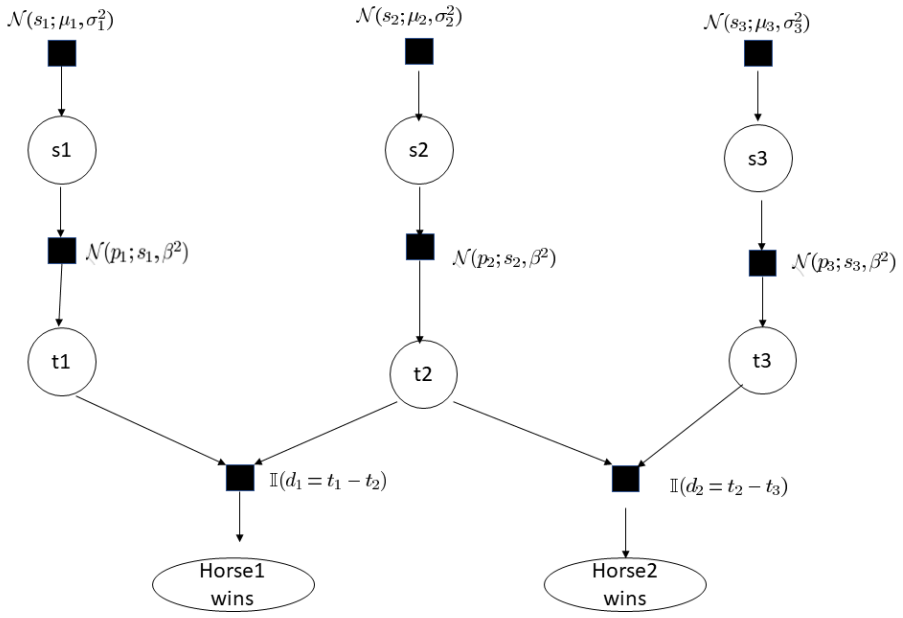


Figure 2. The factor graph describing the joint distribution

In Figure 2, the gray circles indicate the variables, and the black squares show the factor nodes respectively. The joint distribution $P(S, t \mid R, T)$ is computed by the product of all the functions next to the factor nodes. The dependent relationships of the factors are reflected in the graph, and the graph structure is utilized for an efficient inference algorithm.

As we have the joint distribution from the factor graph, we can get back the posterior distribution of the skill level of horses given R and T $P(S \mid R, T)$ by integrating the team performances t_i which is the same as the individual horse performances,

$$P(S \mid R, T) = \int_{-\infty}^{\infty} P(S, t \mid R, T)\, dt. \qquad (10)$$

In the factor graph, the results at the bottom will be used for the update in the approximate message passing part, and the update equations for each section are shown in Figure 3.

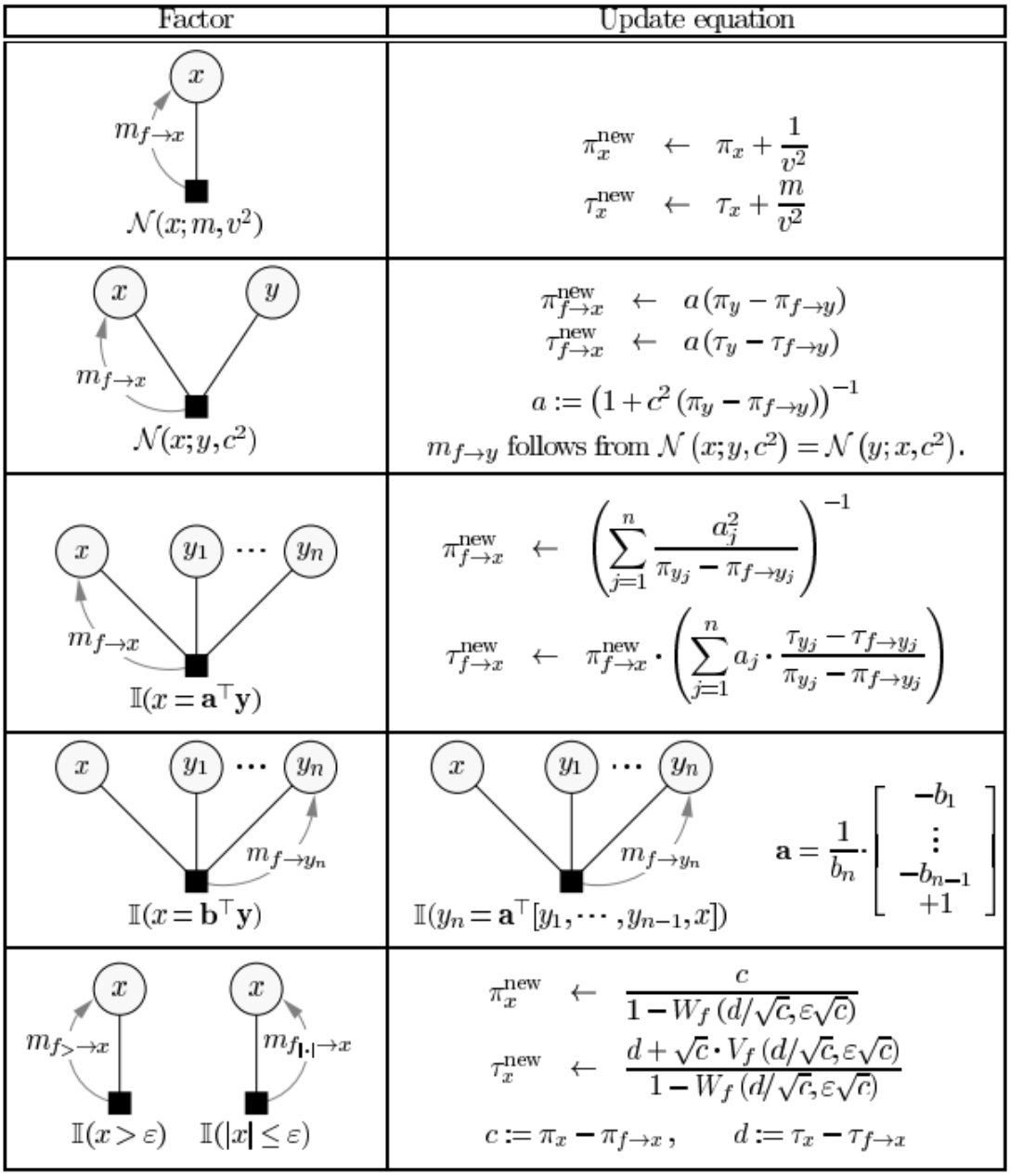


Figure 3. The update equations for the factor graph [23]

2.1.3 Elo-MMR rating system

The Elo-MMR rating system [24] is a novel Bayesian rating system that can be applied to multiplayer competitions with distinct ranks. In order to analyze and quantify the skill levels of horses, all ranking records of horses in the past races are aggregated together, and stronger horses that won consistently in the past will have a higher skill level. The experiments shown in the original paper give a more accurate result with a very efficient time complexity than the existing rating systems when the number of players is large enough.

The Elo-MMR rating system is designed with clear goals. The first goal is to estimate accurate results in a time-efficient manner even though the size of the population is large. The second goal is to be incentive compatible. It means that horses' ratings should not have opposite changes to their race performance. For example, the horse's rating should not be escalated if it gets a place lower than it got in the last race or vice versa. The third goal is to provide a human interpretable rating that the overall skill of a horse can be encapsulated with a single number. One of the reasons for setting the above goals is to avoid complex mechanisms like the message passing in the TrueSkill rating system, which takes more time because the message passing process needs to iterate until convergence has no rigorous justification due to the complexity.

Ultimately, the simplicity of the Elo-MMR system enables rigorous analysis of the massive, monotonic, and robust properties, as mentioned in its name. The massive property indicates that the computation time is scaled only linearly with the increasing size of the population. The monotonic property is equivalent to the incentive-compatible property mentioned in its goal, meaning stronger horses are always

expected to have high ratings. The robust property sets a dynamic bound to the change of the horse's rating so that volatile horses have a larger bound than those consistent horses. As a comparison, Elo-MMR should be better than the Trueskill rating system because the Trueskill rating system cannot meet the robustness requirement and intends to achieve the first two properties without rigorous justification.

The races take place sequentially, and we denote the series of races as $t = 1, 2, \ldots, n$. Then, we denote all horses in the race t as H_t. The i-th horse's skill level at race t is a real random variable denoted as $S_{i,t}$. The performance of the i-th horse in race t is denoted as $P_{i,t}$ and it should have a similar value to $S_{i,t}$. We further assume that each horse's performance and skill level should be independent of its skill level.

The ranking of the race t which is described as the evidence E_t would be responsible for the Bayesian updates. As a result, Elo-MMR calculates the skill level of horse i in race t based on the entire ranking history before race t.
According to the above notations, we can write the joint distribution described by Elo-MMR below,

$$P(S, P, E) = \prod_i P(S_{i,0}) \prod_{i,t} P(S_{i,t} \mid S_{i,t-1}) \prod_{i,t} P(P_{i,t} \mid S_{i,t}) \prod_t P(E_t \mid P_t). \quad (11)$$

The above equation includes one prior distribution and three models.

- $P(S_{i,0})$ represents the initial skill level prior.
- $P(S_{i,t} \mid S_{i,t-1})$ represents the skill evolution model with previous skill levels as information.

- $P(P_{i,t}|S_{i,t})$ represents the performance model with current skill level as information.
- $P(E_t \mid P_t)$ represents the evidence model with performances of all participating horses as information. It is an indicator function that equals one if the relative order of performance of all horses in race t P_t is same as E_t. Otherwise, it equals zero.

2.2 Transformer

In horse racing, the winner is believed to be a relatively skillful horse that defeats the other somewhat weaker horses. Therefore, the dependencies between horses should be captured for comparison and prediction instead of independently treating horses in a single race. Our input is a long sequence of information about all horses in a race. We need a model that can handle sequence modeling and dependencies between the information in the input owing to its attention mechanism. Transformer turns out to be a proper network structure fulfilling our requirements and solves our problem more efficiently than the convolutional neural network and recurrent neural network.

2.2.1 Transformer

The transformer [25] has an encoder-decoder structure. The encoder in the transformer converts input sequences of discrete values to an intermediate sequence of continuous values. Then, the decoder makes use of the intermediate sequence to produce the tokens in the output sequence one by one because the previous token in the output is also the input for producing the next token.

2.2.2 Model Architecture

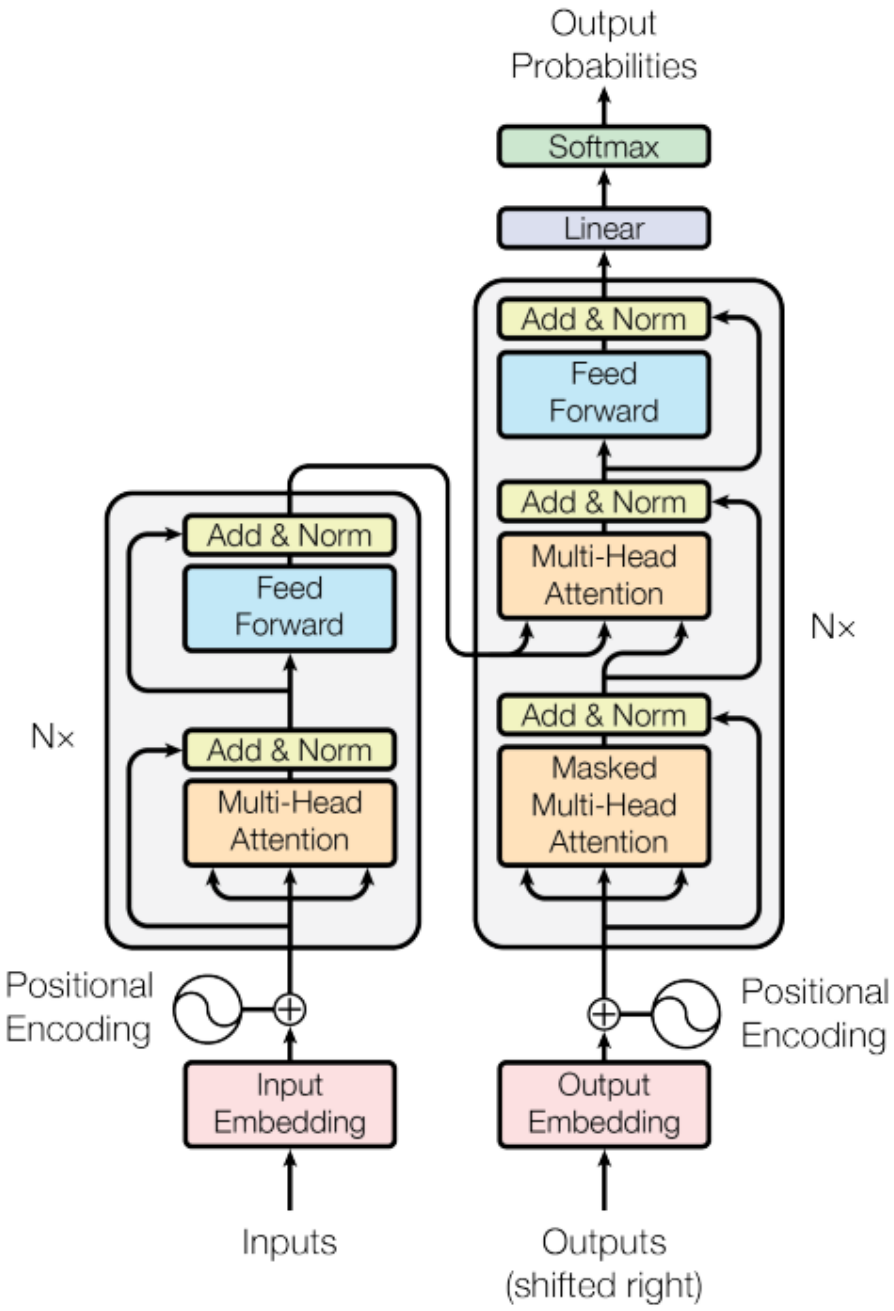


Figure 4. The transformer architecture [25]

In Figure 4, it shows the general structure of a transformer. It contains a stack of self-attention and fully connected layers in the core components encoder and decoder. Details are explained in the later sections.

2.2.3 Encoder

The encoder is formed by N exactly the same layers, while each layer in the stack can be further separated into two sub-layers. The input sequence is first embedded through an embedding layer to have dimension d for each token before entering the encoder stack. The input x of the layer enters the first sub-layer of the encoder stack, which is the multi-head attention mechanism. Then, the original input is added to the output of the multi-head attention mechanism, which is fed to the normalization layer, $LayerNorm(x + mutli - head\_attention(x))$. After that, the outcome of the normalization layer is passed to a fully connected feed-forward layer, and the residual connection is again employed here so that the normalization layer following the feed-forward layer is $LayerNorm(x + feed\_forward(x))$.

2.2.4 Decoder

The decoder is the same as the encoder, except it has additional multi-head attention. A mask is introduced to the first multi-head attention in the decoder stack. The modification aims to prevent positions from attending the unread positions and ensure output at position k can only reference the output before position k.

2.2.5 Attention

In an attention function, the input consists of three vectors: query, keys, and values. Query and keys together undergo a compatibility function to give the weights. Then, the weights are combined with the values to produce the output, a weighted sum of the values.

2.2.6 Scaled Dot-Product Attention

Query and keys both have dimension k, while the values have dimension v. The weights of values are computed by feeding the division of dot products of the queries and keys by the square root of k to a SoftMax function. Generally, the output is generated with the following formula: Q is the matrix of a set of queries, K is the matrix of a set of keys, and V is the matrix of a set of values. The diagram describing the scaled dot-product attention is shown in Figure 5.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{k}}) \cdot V \quad (12)$$

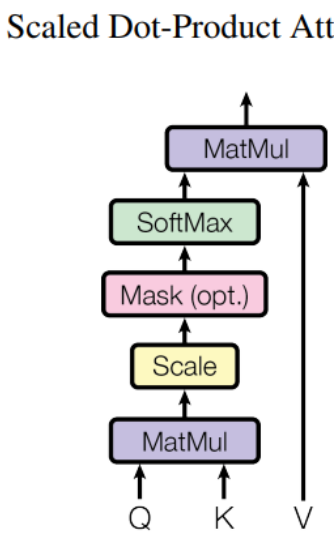


Figure 5. Scaled Dot-Product Attention [25]

2.2.7 Multi-Head Attention

We take an approach alternative to inputting the original queries, values, and keys into the single attention function. The queries, keys, and values of dimension d are linearly projected to h different versions of queries, keys, and values with dimensions k, k, and v respectively. These different versions of queries are parallelly processed with the Scaled Dot-Product Attention. Each of them will produce the values vectors of dimension v. Finally, we concatenate the values outputted from the Multi-head attention, and they are projected as the final values. The following functions mathematically describe the process. The diagram depicting multi-head attention is shown in Figure 6.

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \text{ for i = 1 , … , h, (13)}$$

$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h) \cdot W^O. \text{ (14)}$$

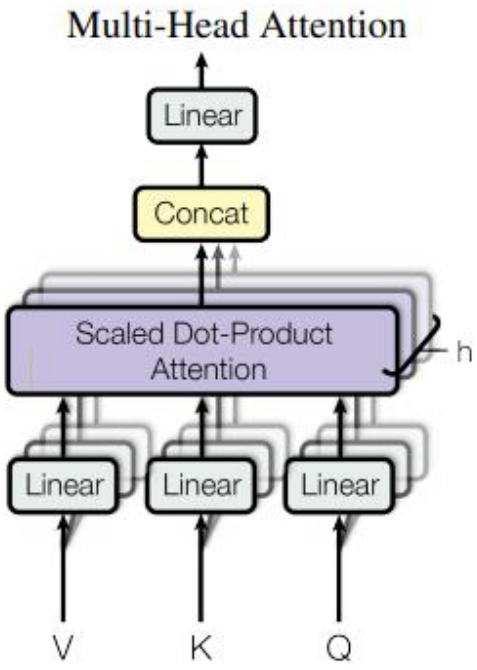


Figure 6. Multi-head Attention [25]

The attention mechanism helps us capture the dependencies between horses because the self-attention layers in the encoder allow each position in the encoder to attend to every position in the former layer of the encoder.

2.2.8 Positional Encoding

The information regarding each information's relative and absolute position in the sequence is inserted because the Transformer does not have recurrence and convolution. Therefore, positional encodings of dimension d are added to the embeddings of the input before it enters the stacks for preserving the ordering and position information. It uses two different functions for encoding the odd and even dimension position.

Let i be the dimension and pos be the position,

$$Postional\_Encoding(pos, 2i + 1) = cos(pos/10000^{2i/d}), \quad (15)$$

$$Postional\_Encoding(pos, 2i) = sin(pos/10000^{2i/d}). \quad (16)$$

2.3 Evaluation Strategies

We want to evaluate the model in the profit-making and accuracy aspects after experiments on the horse racing datasets. We propose the following strategies to decide the performance and effectiveness of adapting the transformer model with the rating of horses as a replacement of win odds in the input.

2.3.1 Random betting (Profit-making Aspect)

We randomly select a horse number from all the participating horses in random betting. If the chosen horse wins, we get back our bet multiplied by the win odd of the winning horse. Otherwise, we lose our chance. It is assumed to be the worst betting strategy because no knowledge is learned from the data before making the prediction.

2.3.2 Lowest Odds betting (Profit-making Aspect)

In Lowest odd betting, we always guess the horse with the lowest win odds as the winner. If the prediction is correct, we gain the amount of bet times the win odd of the winning horse. Otherwise, we lose our bet. It is believed that the lowest odd betting is much better than random betting because the win odds of horses change according to public opinion due to pari-mutuel betting, and thus it reflects the public intelligence. As the public uses their knowledge and experience from the former races in making the prediction, we assume this strategy surpasses random betting.

2.3.3 Multilayer Perceptron Prediction (Accuracy Aspect)

The multilayer perceptron consists of multiple fully connected feed-forward layers is a simple structure of the neural network for making the prediction. No specific

assumption is made about the input properties, and we think it should have lower accuracy than our model.

2.3.4 Transformer without Rating in the Input (Accuracy Aspect)

From the study of previous FYP students, the win odds of horses are the essential features in making the prediction [9]. Since we use the ratings of horses to replace the win odds, we want to show that the ratings are equivalent to the win odds that they could boost the accuracy of the prediction.

Chapter 3

Data Preparation

3.1 Data Collection

Although past horse racing records could be bought directly via websites hosted by companies such as https://horseracedatabase.com/ and https://www.hkhorsedb.com/, which has a database storing the historical data, we prefer to collect the data by ourselves because of the high prices.

In addition to the financial consideration, writing web crawlers to collect data by ourselves provides us more flexibility in the choice of data because we are free to retrieve the data that we want by simply configuring our web crawler. In this project, a web crawler was written for collecting data on the HKJC official websites within a given period. The user can specify the start date and end date so that the crawler will automatically collect the horse race record and horse information from the start date to the end date automatically.

3.2 Data Description

There is a total of 9191 race records in our dataset dated from June 6, 2008 to October, 17 2021. Every row is a race record storing the attributes of a race such as the venue, class, and distance. All races were hosted by the HKJC and took place in Hong Kong. The information about the horses that appeared in the race records was also collected from the HKJC official database.

3.2.1 Racing Record

Table 3 below shows the features of our race record and their detailed information.

Feature	Description	Types	Values
Date	Date of the race	Index	-
Race_id	The id of the race	Index	-
Venue	Location of the race	Categorical	-
Season_race_no	The number of races in the season	Categorical	In range [1 , 800]
Horse_class	Class of the horses Stronger horses compete in high race class	Categorical	1 - 5
Distance	The distance of the race	Categorical	1000, 1200, 1400, 1600, 1650, 1800, 2000, 2200, 2400
Going	Condition of the lane	Categorical	>= 10 distinct values
Course_track	The lane of the race	Categorical	A, A+3, B, B+2, C, C+3
Course_track_code	Description about the lane	Categorical	TURF, ALL WEATHER
Horse_i_place	The rank of horse i in a race	Categorical	14 distinct values
Horse_i_number	The number of horse i in a race	Categorical	14 distinct values
Horse_i_name	The name of horse i	Categorical	> 5000 distinct values
Horse_i_jockey	The name of jockey	Categorical	> 200 distinct value
Horse_i_trainer	The name of trainer	Categorical	> 200 distinct value
Horse_i_actual_weight	The total weight of horse i and gears	Float	-

Horse_i_declared_weight	The weight of horse i	Float	-
Horse_i_finish_time	The time when horse i finishes the race	Float	-
Horse_i_win_odds	The win odd of horse i	Float	-

Table 3. Feature description of race records

3.2.2 Horse Information

Since the horse's information was a helpful indicator of the horse's performance in a race, we gathered 6642 horses that all participated in the races recorded in our dataset for making a comparison between horses in a particular race. Table 4 shows the traits of a horse in our horse dataset.

Feature	Description	Types	Values
Horse_origin	The place of birth	Categorical	>10 distinct values
Horse_age	The age of horse	Categorical	In range [3, 10]
Horse_color	The color of skin	Categorical	>6 distinct values
Horse_sex	The gender of horse	Categorical	Colt, Gelding, Mare etc.
Horse_1st_place_frequency	The frequency of getting 1st place	Categorical	In range [0,20]
Horse_2nd_place_frequency	The frequency of getting 2nd place	Categorical	In range [0,30]
Horse_3rd_place_frequency	The frequency of getting 3rd place	Categorical	In range [0,30]
Horse_total_race	The total count of horse's participation	Categorical	In range [0,100]
Horse_sire	Name of horse's father	Categorical	-
horse_dam	Name of horse's mother	Categorical	-
horse_dam's_sire	Name of horse's maternal grandfather	Categorical	-

Table 4. Feature description of horse records

3.3 Data Analysis

Among all the features describing races and horses, it is believed that not all features are equally important in predicting a horse's performance. Therefore, we would like to study the influences of features on the result of races. In the data analysis, we first investigate the distribution of the selected categorical feature given the winning horses and then look at the likelihood $P(X = x \mid Y = y)$ where x is the selected categorical feature and y is the winning horse. Then, we examine the performance of horses by the correlation between numerical features, especially the finish time and win odds as a horse usually performs well if it finishes the race in a shorter time and has a low win odd. Finally, we want to improve the accuracy of our model and speed up the training process so we carry out feature selection to eliminate unrelated features and noises.

3.3.1 Categorical Features

3.3.1.1 Age

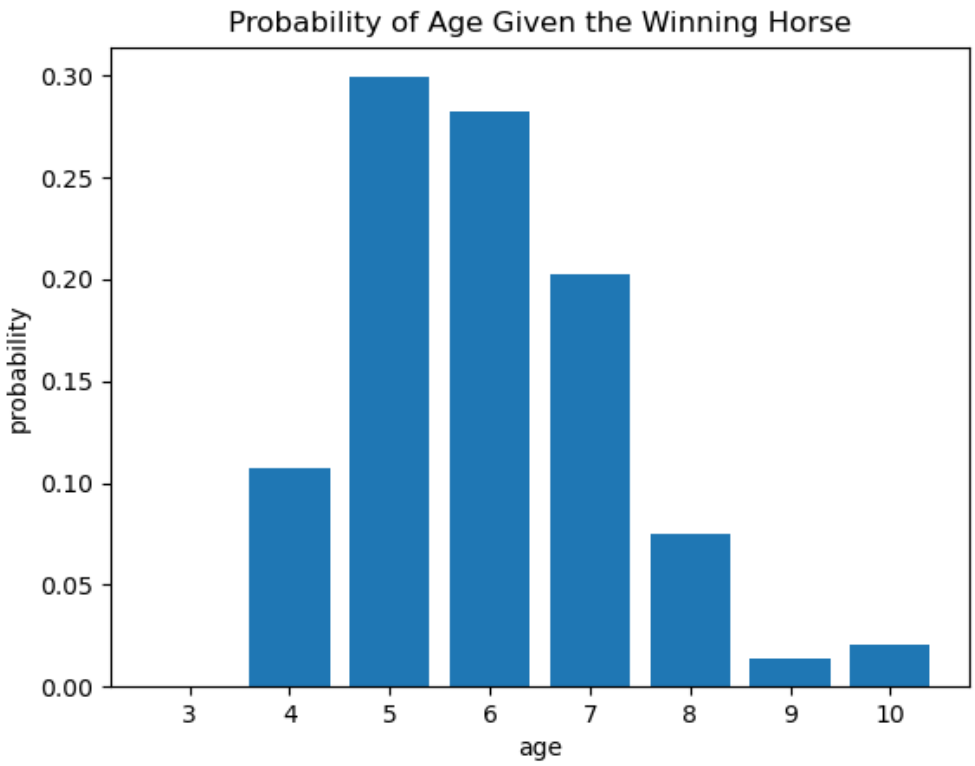


Figure 7. The distribution of age given the winning horse

Our data demonstrate the declining performance of horses with increased age, as shown in Figure 7. Among all the winning horses, more than 50% are horses aged between 5 and 6 as horses' optimal body weight and skeleton are reached at 4 or 5 years old [26]. The number of winning horses decreases substantially after the age of 5. It implies that the overall performance of the majority of horses reaches its peak when they are 5 or 6 years old and then declines due to the decrease in stamina, speed, and power bought to aging. The horses aged between 3 and 4 accounts for approximately 11% of the winning horses. One explanation for fewer winning horses of lower age is that they have not joined enough competitions to be very skillful, and

they are still growing.

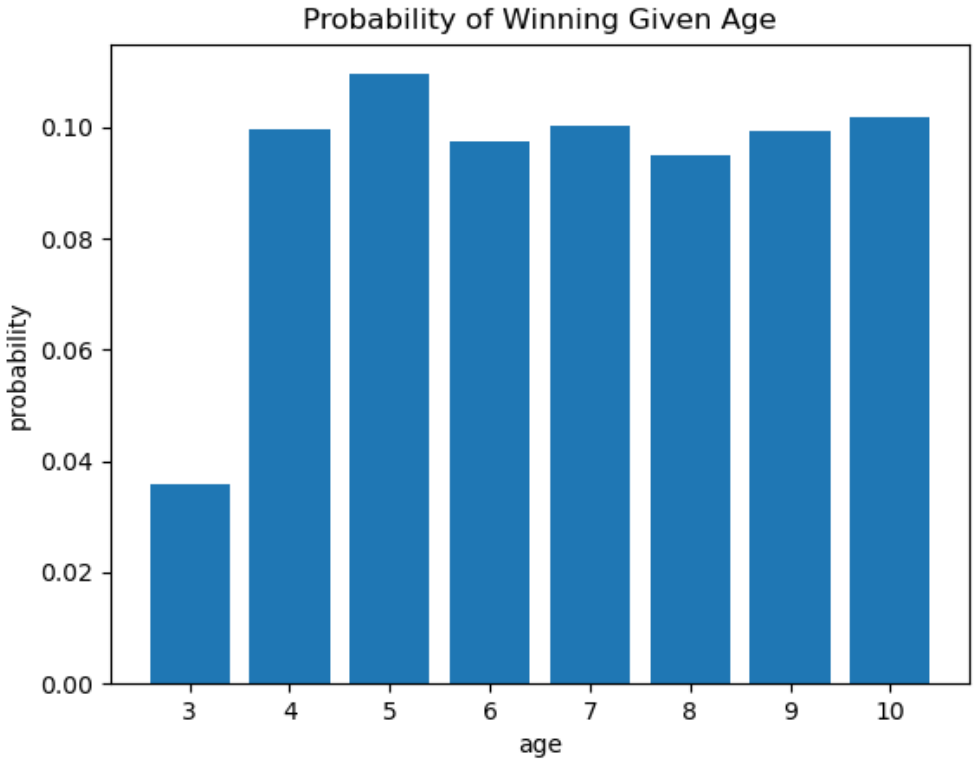


Figure 8. The distribution of winning horses given age

Although the likelihood tends to choose horses aged between 5 and 6 to be winners, we observe that horses have a similar probability of winning at around 10% for all ages except 3, shown in Figure 8. It suggests that the winning condition cannot be determined solely by the age of an individual horse.

3.3.1.2 Origin

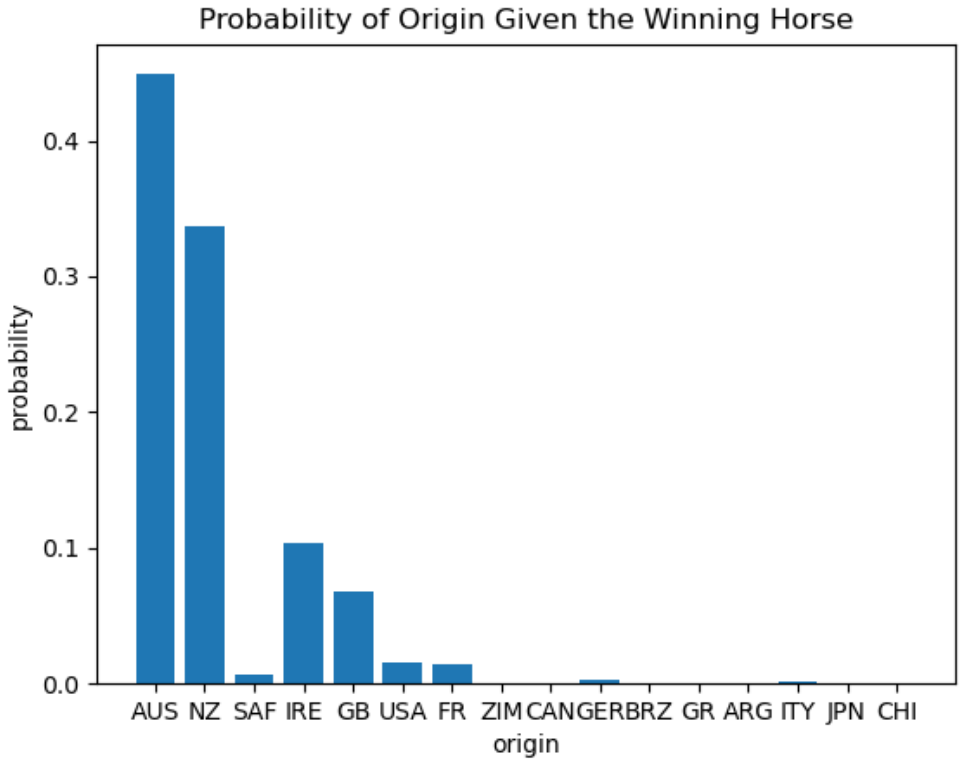


Figure 9. The distribution of origin given the winning horse

Most winning horses were born in Australia or New Zealand, as shown in Figure 9. It reflects that horses born in Australia or New Zealand usually perform better than horses from other countries. This information is useful when we want to do a simple classification to identify all horses with various origins in a single race into two classes that are likely and unlikely to win. In this situation, horses from Australia or New Zealand will be classified as possible to win, while horses from other countries will be classified as unlikely to win.

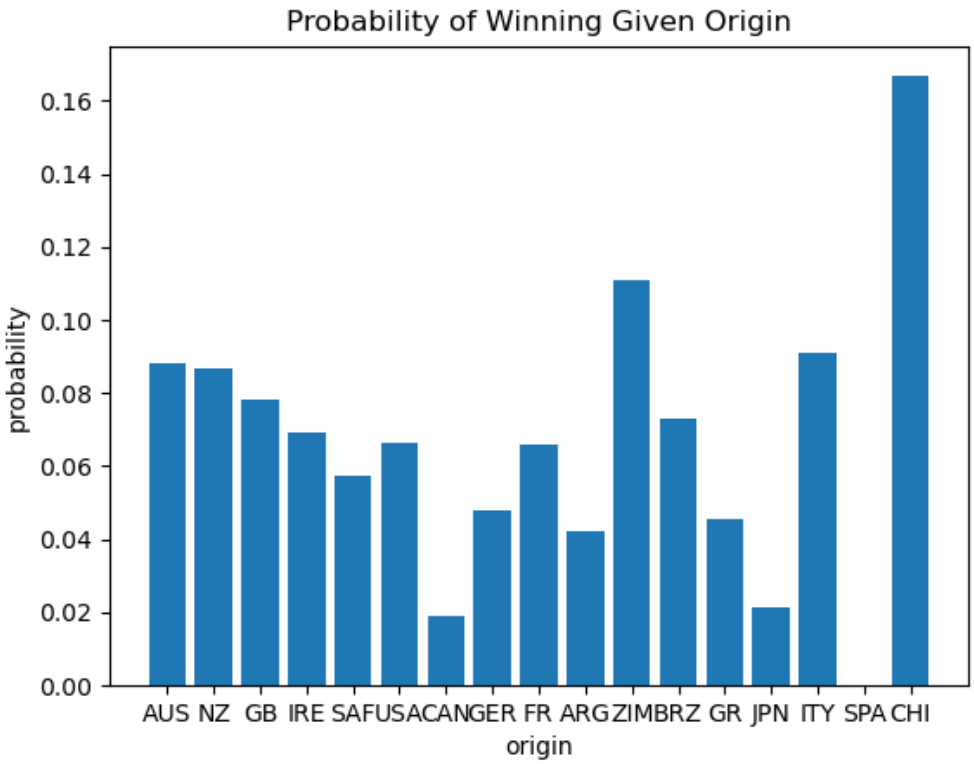


Figure 10. The distribution of the winning horse given origin

Since most horses in horse races are imported from Australia and New Zealand, this may bias the winning distribution that horses from Australia and New Zealand are usually winners. After conditioning the winning probability by the origin, we see horses from the Republic of Zimbabwe and República de Chile. Nevertheless, the number of horses coming from the Republic of Zimbabwe and República de Chile is tiny, while the number of horses from Australia and New Zealand is huge. Figure 10 shows that horses Australia and New Zealand are still likely to be the winner in the actual case compared to other countries except for the Republic of Zimbabwe and República de Chile.

3.3.1.3 Color

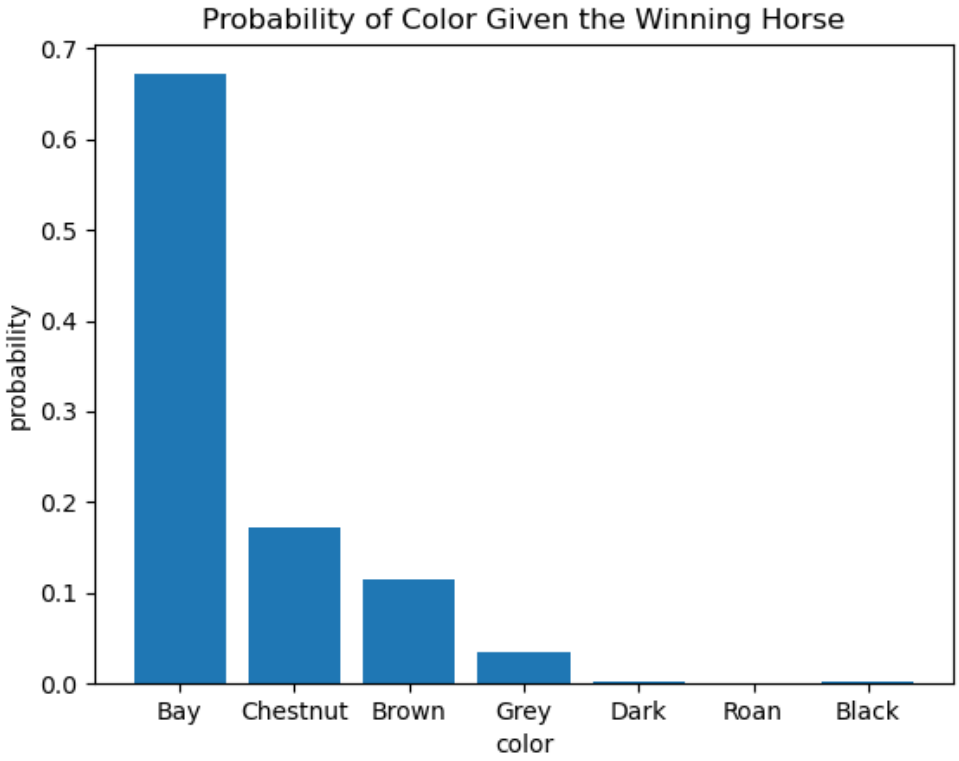


Figure 11. The distribution of color given the winning horse

More than 65% of winning horses have skin color Bay as shown in Figure 11. The second most color is Chestnut with 17%. The remaining colors like Brown, Grey, Dark, Roan, and Black only constitute a small portion of the winning horse. The extensive distribution of color Bay in the winning horse suggests that color would be a good choice for being the early decision boundary in machine learning methods that adopt the greedy approach such as decision tree.

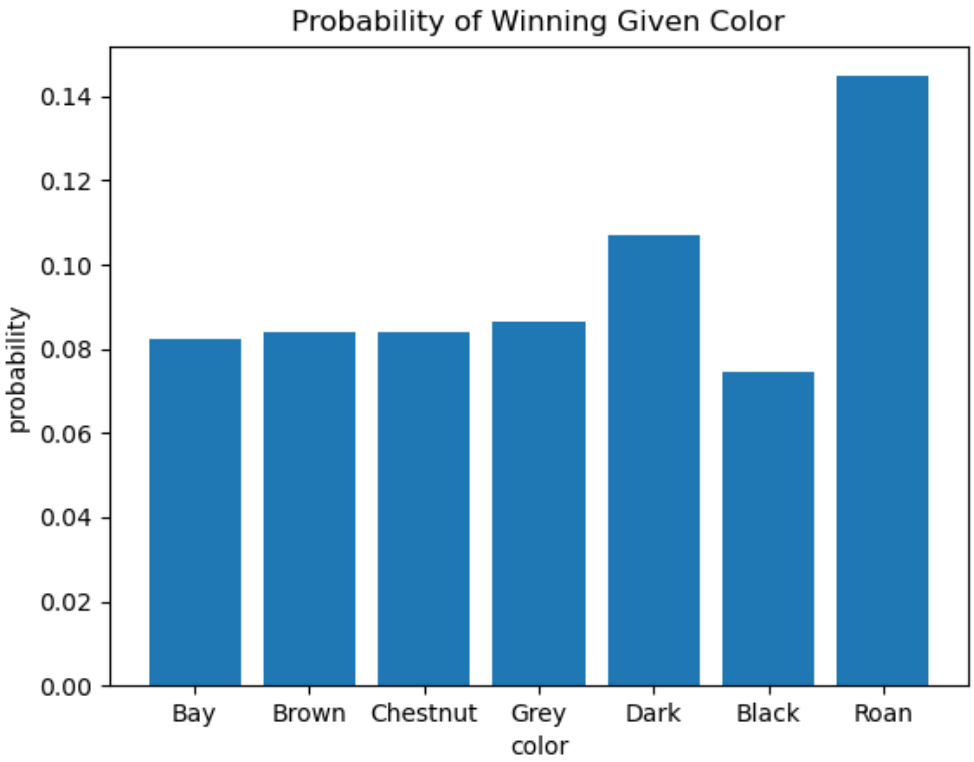


Figure 12. The distribution of the winning horse given color

When the winning probability is conditioned on the color, the advantage of horses with the color Bay loses while those colors which are less likely to appear in winning, such as Dark and Roan horses surpass. Also, the winning probability of a horse with the color Bay is the second-lowest in Figure 12, and it implies that our observation from Figure 11 is biased as a large portion of horses in horse races have skin color Bay.

3.3.1.4 Sex

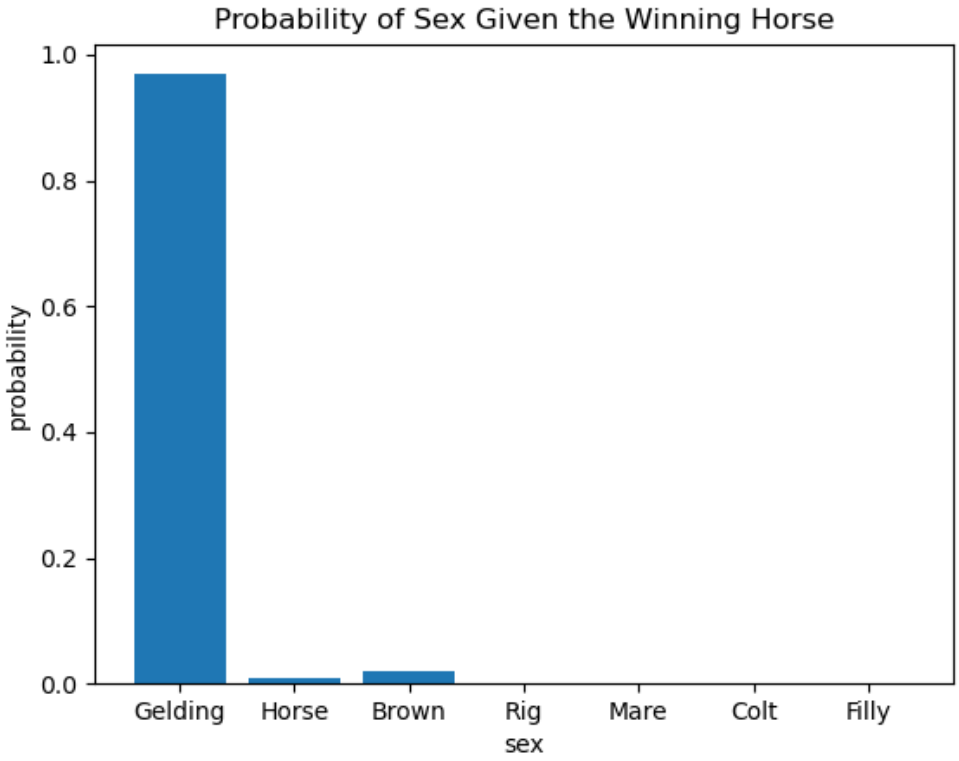


Figure 13. The distribution of sex given to the winning horse

The sex Gelding dominates the likelihood distribution of sex in winning horses. Over 97% of winning horses with sex Gelding, as shown in Figure 13. The sex Horse and Brown only constitute a tiny portion of the winning horse with approximately 3% in total. This likelihood is highly biased because almost all horses in a horse race have Gelding, and therefore this feature should have extremely few impacts on the race result.

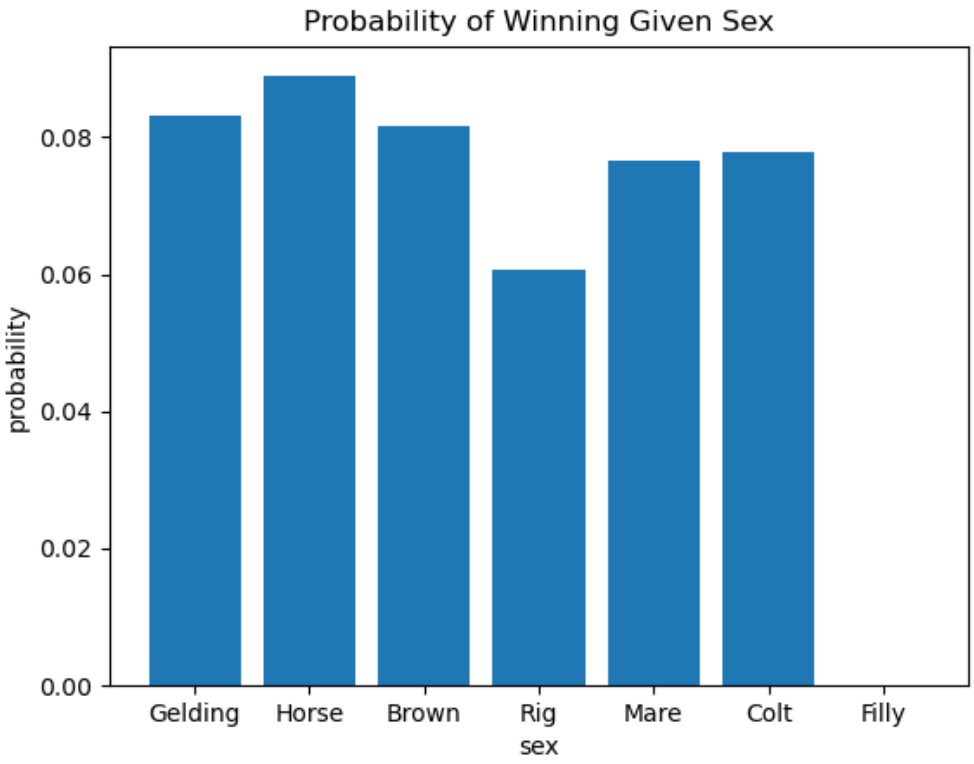


Figure 14. The distribution of winning horse given sex

The conditional probability of winning given the sex of the horses confirms our assumption that the horse with the sex Gelding is the most likely the winner is flawed because the chance of winning given the sex is Gelding has a similar value to the probability of winning given other sex. From Figure 14, we are more confident that the horse with sex Horse will win the race as it has the highest conditional probability of winning among all horses of other sex.

3.3.1.5 Draw

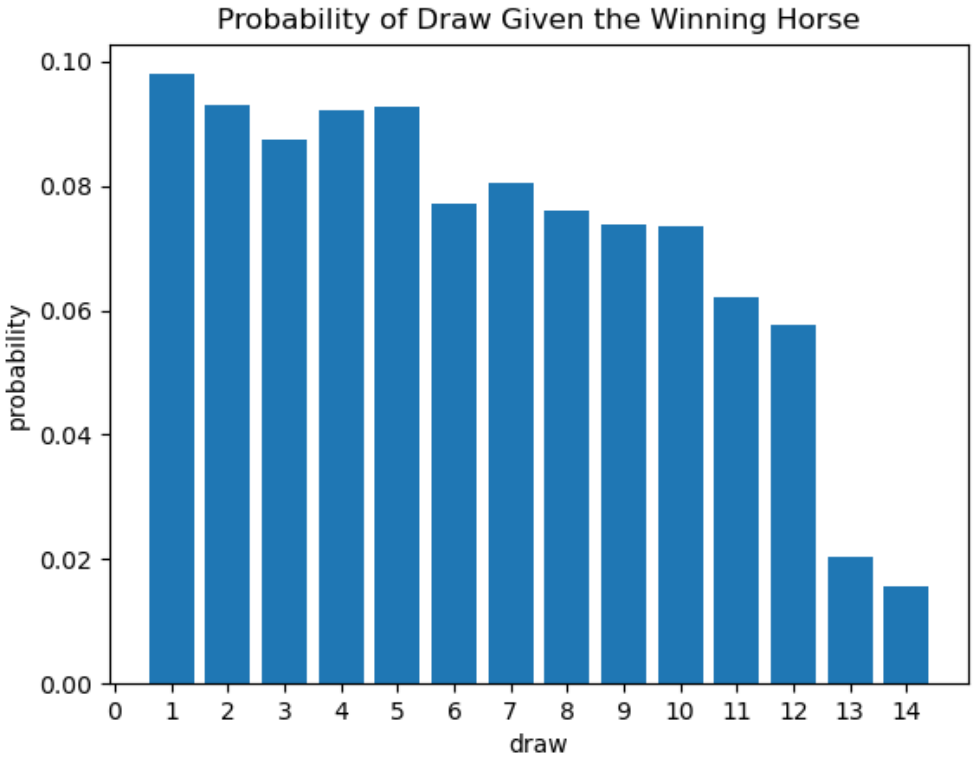


Figure 15. The distribution of draw given the winning horse

Horses with smaller draw numbers are considered to be opportune in horse racing because they are arranged towards the center of the circular track, as shown in Figure 15. The running distance of those horses is thus relatively shorter than horses with more significant draw numbers, which means horses with smaller draw numbers need a shorter time in finishing the race. Our data agrees with our assumption about the advantage of smaller draw numbers since there is a declining proportion of winning horses with increasing draw numbers.

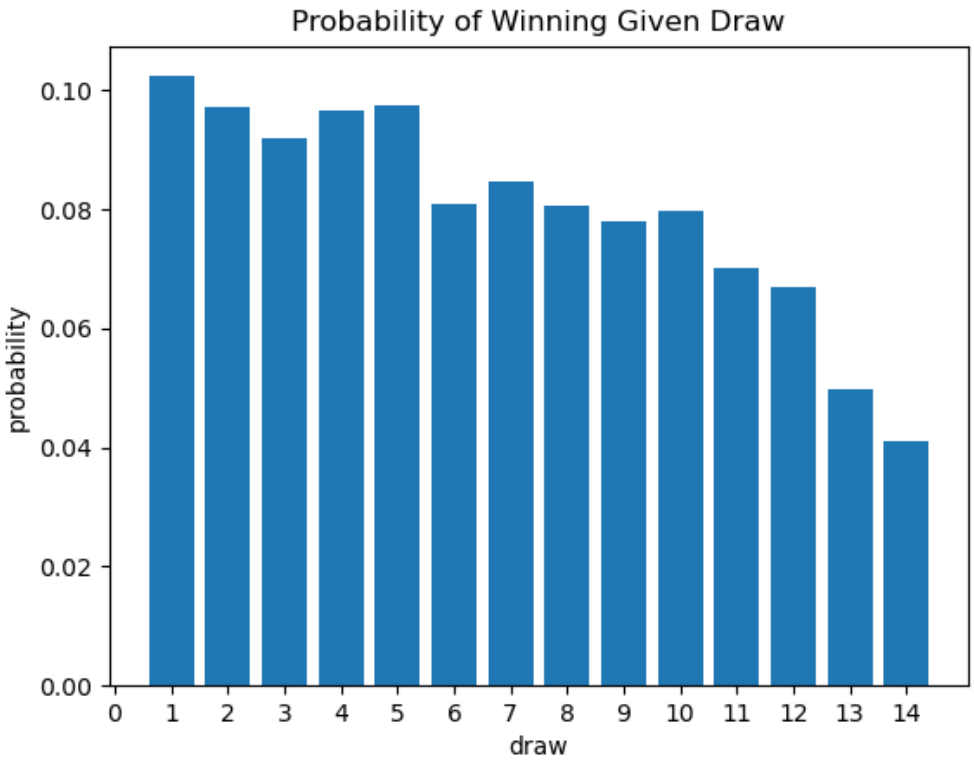


Figure 16. The distribution of the winning horse given the draw

The conditional probability of winning given the draw has a similar shape to the possibility of draw given the winning horses as shown in Figure 16. Hence, the fact that the horses with a smaller draw number are more likely to be the winner is assured. However, the horses with a large draw number also win in some races so the other factors should be considered in determining their winning probability.

3.3.2 Numerical Features

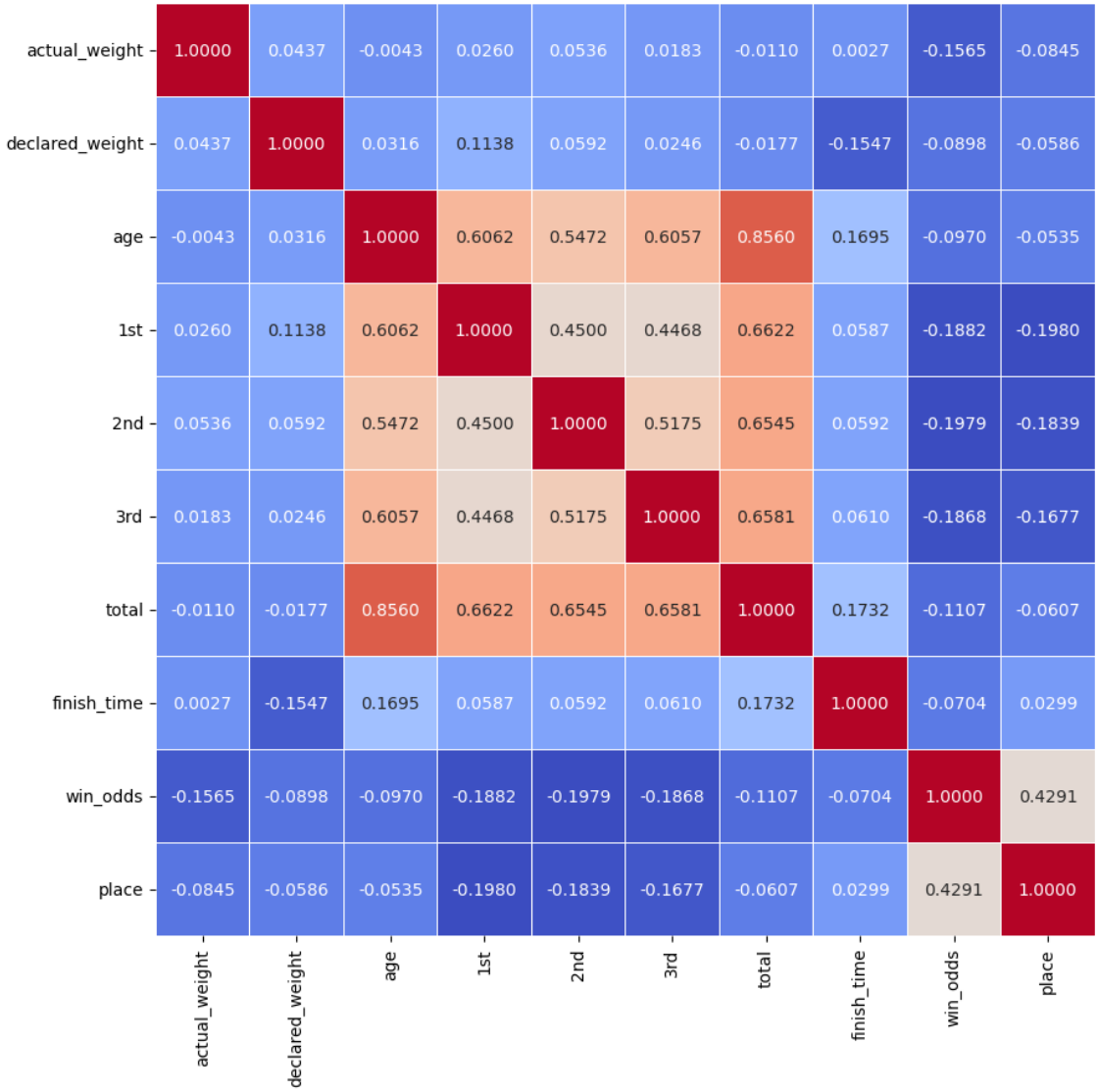


Figure 17. The correlation matrix of numerical features

In analyzing the numerical features, we investigate the correlation between each pair of horse features in our dataset from Figure 17. The cell of darker color in the correlation matrix implies a stronger correlation or vice versa. Some essential horse features that strongly influence the result of a race are selected for the following discussion.

3.3.2.1 Frequency of 1st Place

We examine the row of feature frequency of 1st place, which counts as a winner in a race. It has a significant correlation with the frequency of 2nd place and 3rd place, which are 0.4500 and 0.4468 respectively. As the correlation coefficients are positive, it infers the positive relationship between the frequency of 1st place, 2nd place, and 3rd place. The relationship matches our expectation that a horse with good performance in the past, getting first three places in the past race, often performs well in the next race. Notice the negative correlation of -0.1882 between the frequency of 1st place and the win odds. The public has the same opinion about the consistent performance of horses in future races, so they tend to bet the horse with a large count of 1st place, and it results in a lower win odds of the horse owing to the pari-mutuel betting system. Besides, the consistent performance is proven by the negative correlation of -0.1980 between the count of 1st place and the horse's places in races.

3.3.2.2 Finish Time

The finish time measures horse performance since we assume a stronger horse will finish a race in a shorter time. This motivates us to examine the correlation between the finish time and other horse features. A negative correlation of -0.1547 between finish time and declared weight is shown in the correlation matrix. The handicapping policy by the HKJC adds weights to well-performed horses, and the declared weight is increased so that the chances for horses of worse performance are increased [11]. Our data shows that the policy is not effective enough because the well-performed horses with more declared weights still have a shorter finish time. On the other hand, there is a positive correlation of 0.1695 between the finish time and the age. This

agrees with our analysis of the age of horses that the performance of horses declines with age.

3.3.2.3 Win Odds

The win odds of horses reveal the general guess of the public since the win odds of horses change with the amount of bet. The more popular the horse, the lower the win odds of the horse. From the correlation matrix, win odds have negative correlations of -0.1882, -0.1979, and -0.1868 between the frequency of 1st place, 2nd place, and 3rd place respectively. This implies that the frequency of 1st place, 2nd place, and 3rd place guides the public to decide. The larger the number of this statistic, the lower the win odds. Another discovery is the positive correlation of 0.4291 between the win odds and the place. This shows that public intelligence is accurate in some sense. For example, if the public does not think the horse will win, they will not bet on it, and the hose will have very high win odds. If the public intelligence is accurate enough, the horse with a high win odd should not perform well and get a small number.

3.4 Data Preprocessing

The raw data scraped from websites are not clean and well organized, so they should be preprocessed into a desirable format before feeding them into our neural network models. We had done the following four steps data imputation, data encoding, input normalization, and rating generation on our dataset before starting the experiments.

3.4.1 Data Imputation

In the data collection process, we inevitably encounter network errors such as link rot or an unresponsive server, especially when the target data is old. This happened when we collected the data of some retired horses that took part in the races before 2010. Hence, our data set is missing a small part of horse data about those retired horses. However, omitting or removing the horse records with missing information is unadvisable. The records may affect the quality of the knowledge extraction procedure, and biased estimation would be made when doing the analysis [27].

Addressing the missing information, we decided to do data imputation on our dataset using the k nearest neighbors method. First, we extract all complete horse records without missing values. Then, we place the missing value in an incomplete record by looking for its k nearest neighbors in the complete horse records. The value filled in the missing part will be the mean of neighbors if the type of feature is numerical. Otherwise, we do a majority vote on the neighbors and place the most common categorical value in the missing part [27].

Instead of implementing the k nearest neighbors, we invoked the KNN Imputer from the Scikit Learn library to ensure simplicity and correctness.

3.4.2 Data Encoding

The input of our neural network models must be numerical, but some of our data are categorical. For instance, the horse's name, jockey name, distance, and course track are categorical values. For this reason, we need to transform our categorical data into numerical by data encoding.

A straightforward method is converting the categorical data in the form of one-hot encoding in which we use k binary features to represent a categorical feature of 2^k classes. The value of the binary feature is either 1 or 0. However, the dimension of our input will be increased drastically for representing all categorical data, and it requires extra memory and more computational time for the training [28]. The principal component analysis is a possible solution to the expansion of dimension caused by one-hot encoding because it can reduce the data dimension while preserving the variance of data points.

The ordinal Encoding scheme is also a good option for our data as we do not want additional memory usage and extra computational time due to the one-hot encoding. In this scheme, a unique integer means a category, and no new columns are added, so the data dimension is the same as the original. Furthermore, the order of ordinal variables is preserved in this scheme [29]. For example, the feature place has 14 classes representing the ranks of horses in a race. We encode the horses of higher rank with a smaller integer to preserve the ranking order.

3.4.3 Normalization

Normalization of the input was done before the training of our models. It was shown that the normalization of input data can produce a better result and speed up the training process. On the one hand, values of all variables are scaled to have the same range, which saves the effort for backward propagation in changing the weight of variables. On the other hand, the same scale of all variables balances the focus of error minimization in the weight correction algorithm, so that importance of variables is distributed evenly to avoid bias [30].

We use z-score normalization which takes the mean and standard deviation of each feature in the column direction of our input vector and uses that information to compute the values for the corresponding feature. The formula is shown below,

$$x_i' = \frac{x_i - \bar{x_i}}{\sigma_i}. \tag{17}$$

where x_i' is the computed value, $\bar{x_i}$ is the mean of the feature and σ_i is the standard deviation of the feature.

3.4.4 Rating Generation

Rating the horse's performance is one of the focuses of this project. Nonetheless, the ratings mentioned in the methodology do not exist on the HKJC websites, and we need to calculate those ratings with the information provided by our dataset.

In rating generation, we mapped horse records to race records under the guidance of the horse names in race records so that we obtain the race records with horse records ordered from 1st place to the last place. Then, we reformatted each record into a JSON file and named each file with a number. The smaller the number, the older the race. After that, we invoked a rating computation library [31] and used all JSON files as the input. Then we had a list of horses with their rating in each race which was then merged into our original dataset. The list of JSON files is shown in Figure 18. The content of the rating file for each horse is shown in Figure 19.

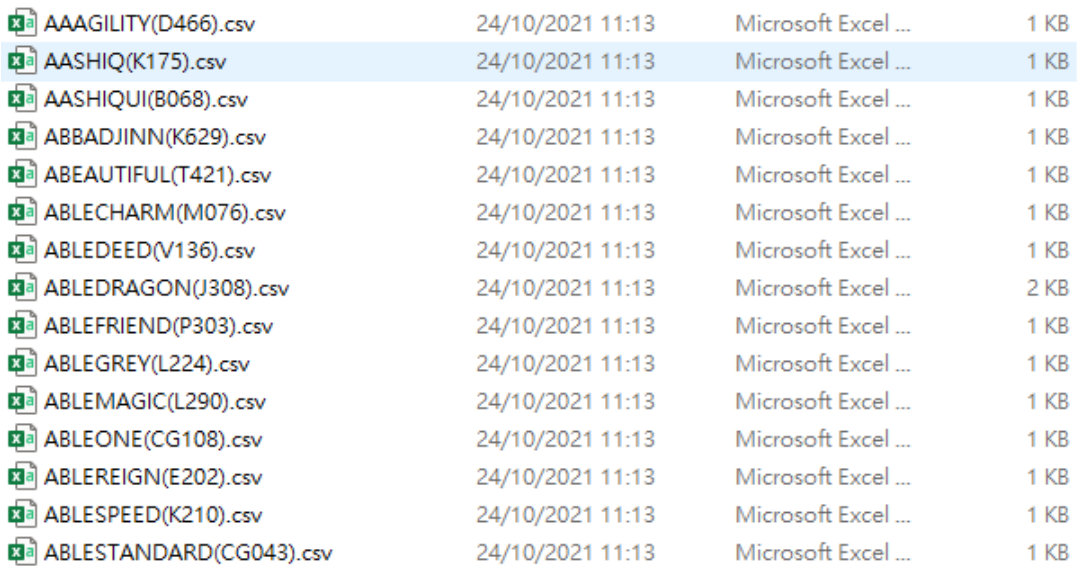


Figure 18. The list of JSON files for rating computation

	A	B	C	D	E
1	contest_index	rating_mu	rating_sig	perf_score	place
2	8315	1329	174	1294	13
3	8372	1373	133	1421	12
4	8444	1337	114	1270	14
5	8515	1336	102	1332	12
6	8586	1305	95	1208	14
7	8704	1283	90	1211	11

Figure 19. The rating of the horse

3.5 Feature Selection

As the data dimension is large, the number of parameters increases, and the time for training parameters in the model is expensive. Furthermore, irrelevant features in the data are noises that cause the model to consider irrelevant information and harm the neural network's performance [32]. Feature selection is used to reduce the dimension of the data so that a small but sufficient subset of features becomes the input of the model to achieve a shorter learning time and higher accuracy.

3.5.1 Random Forest

A measure of the feature relevance is needed to filter out unnecessary features with low relevance scores. The random forest classifier combined with the Gini index could estimate the feature relevance because the changes of Gini impurities due to a feature indicated the importance of the feature [33]. The higher the increment in the leaf’s purity, the more the relevance of the feature. This enables an explicit feature elimination based on the changes in the leaf’s purity and results in using a small subset of significant features only.

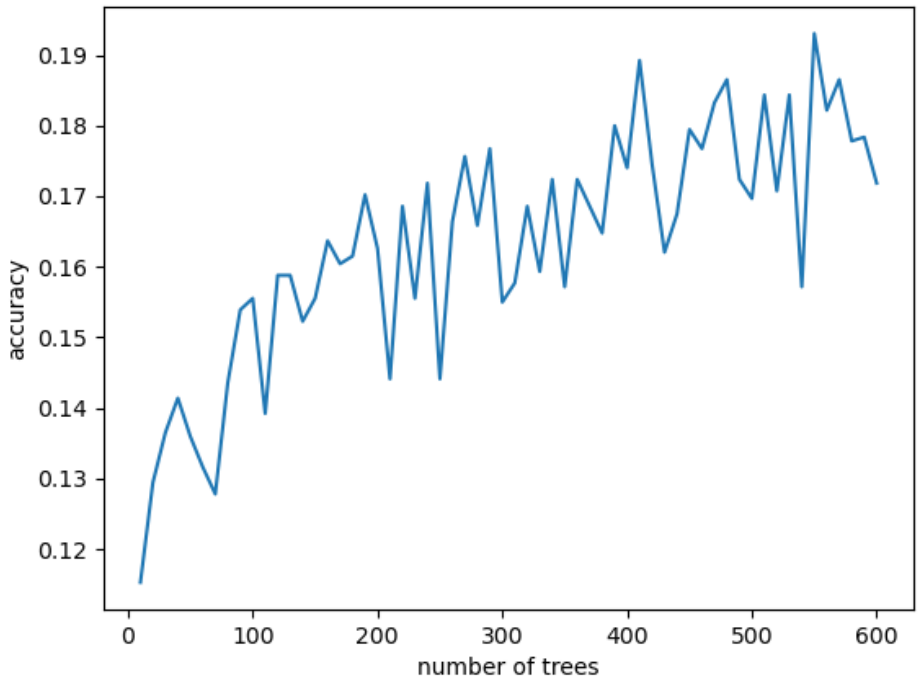


Figure 20. Accuracy versus Number of Trees in Random Forest

The random forest classifier must be trained first before viewing the Gini feature importance associated with it. However, the accuracy of the random forest classifier varied with the number of trees, and we believe that the classifier with the highest accuracy could provide a more precise estimation of the feature relevance. In Figure 20, the random forest classifier has the highest accuracy of 19.3% when there are 560 decision trees, and the Gini feature importance of this classifier is picked to be an indicator of feature relevance.

3.5.2 Importance of Features

	importance
horse-1-rating_mu	0.007217
horse-5-rating_mu	0.007188
horse-2-rating_mu	0.007157
horse-6-rating_mu	0.007117
horse-4-rating_mu	0.007095
horse-8-rating_mu	0.007063
horse-3-rating_mu	0.007062
horse_7_declared_weight	0.007011
horse_4_declared_weight	0.006998
horse-7-rating_mu	0.006983
horse_2_declared_weight	0.006965
horse_1_declared_weight	0.006961
horse_6_declared_weight	0.006954
horse-9-rating_mu	0.006940
horse_5_declared_weight	0.006910
horse_3_declared_weight	0.006871
horse_8_declared_weight	0.006863
horse-10-rating_mu	0.006823
horse_9_declared_weight	0.006781
horse_3_total	0.006732
horse_10_declared_weight	0.006685
horse_8_total	0.006648
horse_5_total	0.006626
horse_2_total	0.006593
horse_6_total	0.006592
horse_4_total	0.006561
horse-12-rating_mu	0.006530
horse_1_total	0.006491
horse_7_total	0.006462
horse_10_total	0.006438
horse-11-rating_mu	0.006435
horse_12_declared_weight	0.006413
horse_11_declared_weight	0.006305
horse_9_total	0.006238
horse_11_total	0.006147
horse_12_total	0.005923
horse_1_1st	0.005853
horse_8_actual_weight	0.005482
horse_2_1st	0.005480
horse_9_actual_weight	0.005452
horse_6_actual_weight	0.005385
horse_11_actual_weight	0.005348
horse_7_actual_weight	0.005321
horse_4_actual_weight	0.005297
horse_3_1st	0.005284
horse_5_actual_weight	0.005284
horse_4_1st	0.005279
horse_5_1st	0.005267
horse_8_1st	0.005266
horse_7_1st	0.005264
horse_10_actual_weight	0.005210
horse_3_actual_weight	0.005168
horse_6_1st	0.005005
horse_2_actual_weight	0.004981
horse_9_1st	0.004961
horse_12_actual_weight	0.004893
horse_1_3rd	0.004793
horse_1_2nd	0.004791
horse_10_1st	0.004787
horse_3_2nd	0.004785
horse_2_2nd	0.004783
horse_4_3rd	0.004754
horse_5_2nd	0.004709
horse_2_3rd	0.004702
horse_4_2nd	0.004698
horse_7_3rd	0.004689
horse_6_2nd	0.004681
horse_6_3rd	0.004660
horse_5_3rd	0.004653
horse_3_3rd	0.004639
horse_8_3rd	0.004604
horse_7_2nd	0.004563
horse_9_3rd	0.004552
horse_8_2nd	0.004550
horse-6-rating_sig	0.004544
horse_9_2nd	0.004527
horse_10_3rd	0.004499
horse_11_1st	0.004495
horse_12_1st	0.004450
horse-5-rating_sig	0.004422
horse-1-rating_sig	0.004406
horse-9-rating_sig	0.004360
horse_10_2nd	0.004348
horse-2-rating_sig	0.004331
horse-7-rating_sig	0.004321
horse-3-rating_sig	0.004307
horse-8-rating_sig	0.004282
horse_11_3rd	0.004248
horse-4-rating_sig	0.004246
horse-10-rating_sig	0.004194
horse-11-rating_sig	0.004161
horse_11_2nd	0.004158
horse_12_3rd	0.004129
horse_12_2nd	0.004081
horse-12-rating_sig	0.004035
horse_1_actual_weight	0.003719
horse-13-rating_mu	0.003105
horse-14-rating_mu	0.003031
horse_13_declared_weight	0.003014
horse_13_total	0.002968

Figure 21. Gini Feature Importance

The Gini feature importance shown in Figure 21 is extracted from the random forest classifier and arranged in descending order. Ratings, declared weights, the total number of winning a race, and the frequency of getting the first places are essential

among all the features in the input data, and they are candidates to be kept in the feature selection.

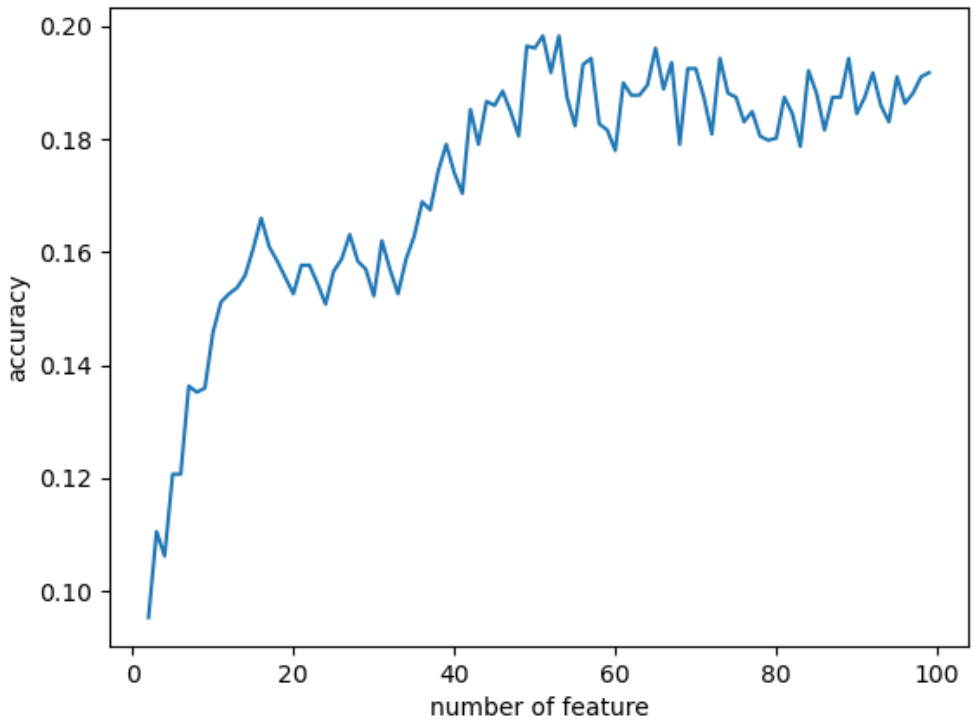


Figure 22. Accuracy of Random Forest with n Most Important Features

Determining the number of the most important features kept for in the input of our neural network, we investigate how the accuracy changes with a varying number of features. In Figure 22, The random forest classifier reaches its maximum accuracy when the number of features is around 50, and the accuracy remains more or less the same after it. So, only the 50 important features are kept in the input of the neural network, and other features are eliminated if we want to speed up the training or lack memory in training.

Chapter 4

Methodologies

4.1 Overview

The win odds capture the relative expected performance of horses in a race because bettors tend to bet on a relatively more robust horse, and the pari-mutuel betting setting will therefore decrease the win odds of the horse with better-expected performance. Moreover, previous final year project, students in LYU1805 illustrated the significance of win odds in horse race prediction [10]. Finish time is also an important metric to evaluate the relative performance of horses since stronger horses can finish the race in a shorter time. However, both win odds and finish time should be excluded from the feature list because we cannot obtain accurate win odds until the start of the race owing to the dynamic nature of win odds, and we do not know the finish time of horses when we predict new races. Therefore, we decide to find another metric to help us figure out the relative performance of horses.

Rating systems estimate the relative skill level of horses based on their historical performance. As we can quickly assess a horse's past racing record from the HKJC website, we apply rating systems here to calculate the relative skill point of horses, and we wish the ratings could replace the effect of win odds in our prediction.

Rating systems have different underlying assumptions in calculating the relative skill point. We want to see which rating system best represents the relative skill point of horses, so we will experiment with three different rating systems and find the one that

produces the best result.

Besides the relative skill level of horses, win odds also rely on the dependencies between horses' attributes. The attributes of horses in a race are not independent in our context, and thus the probability of winning for each horse should be conditioned on the attributes of all participating horses. Simple models such as linear regression and a single decision tree are not suitable for this problem because the relationships between the attributes of horses are sophisticated and cannot be easily captured by these simple models. So, models that can learn complex non-linear relationships and are dedicated to referencing all attributes of horses in estimation should be selected for the prediction. Multilayer perceptron and transformer are chosen to be the models for this consideration.

Therefore, we will experiment with the combinations of rating systems and the selected neural network architectures to see whether they can compensate for the exclusion of win odds in prediction. Then, we will compare the results of different combinations and evaluate their performance by using the evaluation strategies proposed in section 2.3.

A customized embedding scheme named Horse Token embedding scheme is designed to encapsulate the horse attributes better and improve the model accuracy in the second stage of the project. The impact of the embedding scheme on the prediction is evaluated by comparing the preciseness of the predictions before and after applying the embedding scheme to the best model selected through the model comparison in the first stage of the project. The methodology diagram is shown in Figure 23.

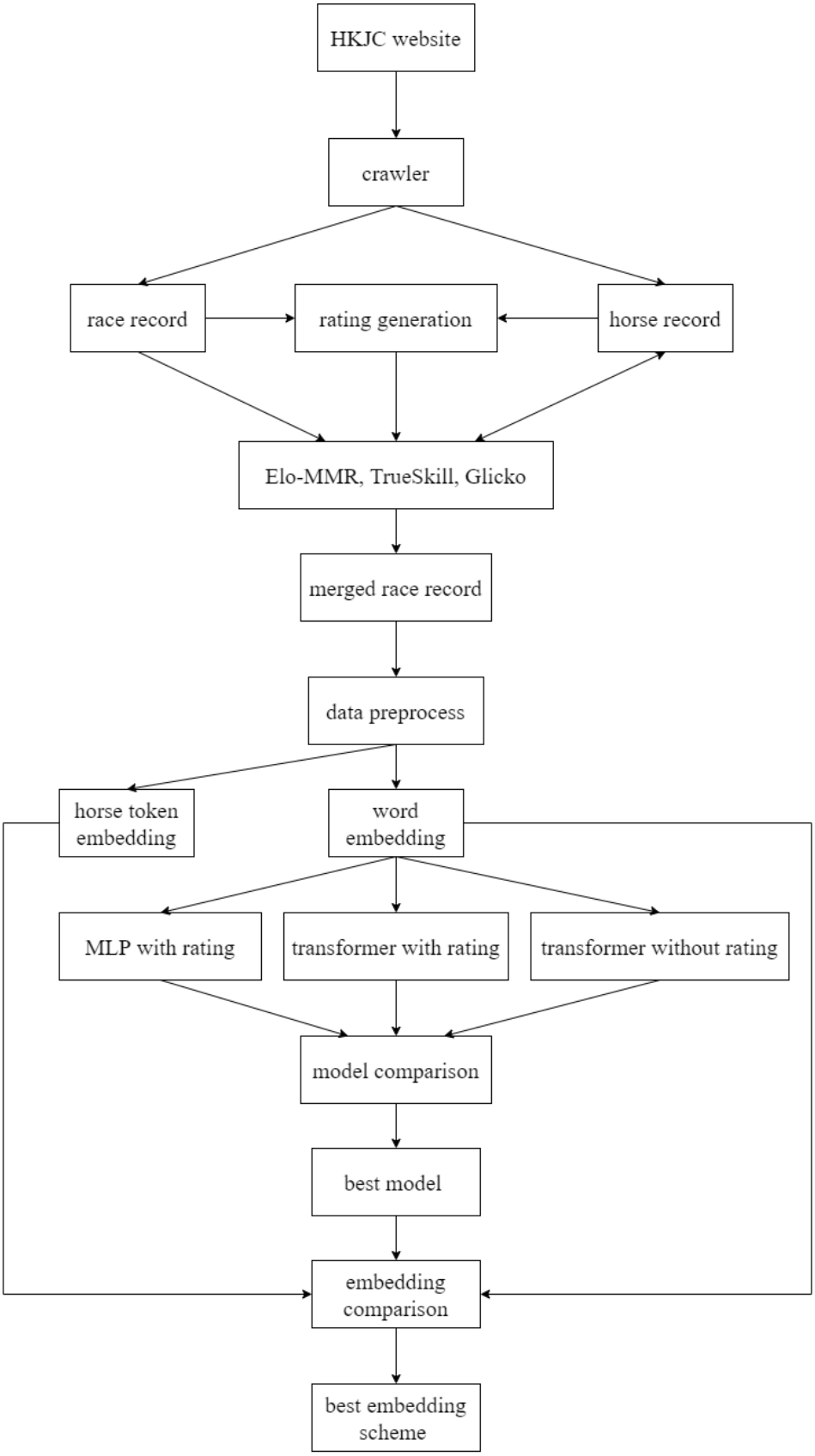
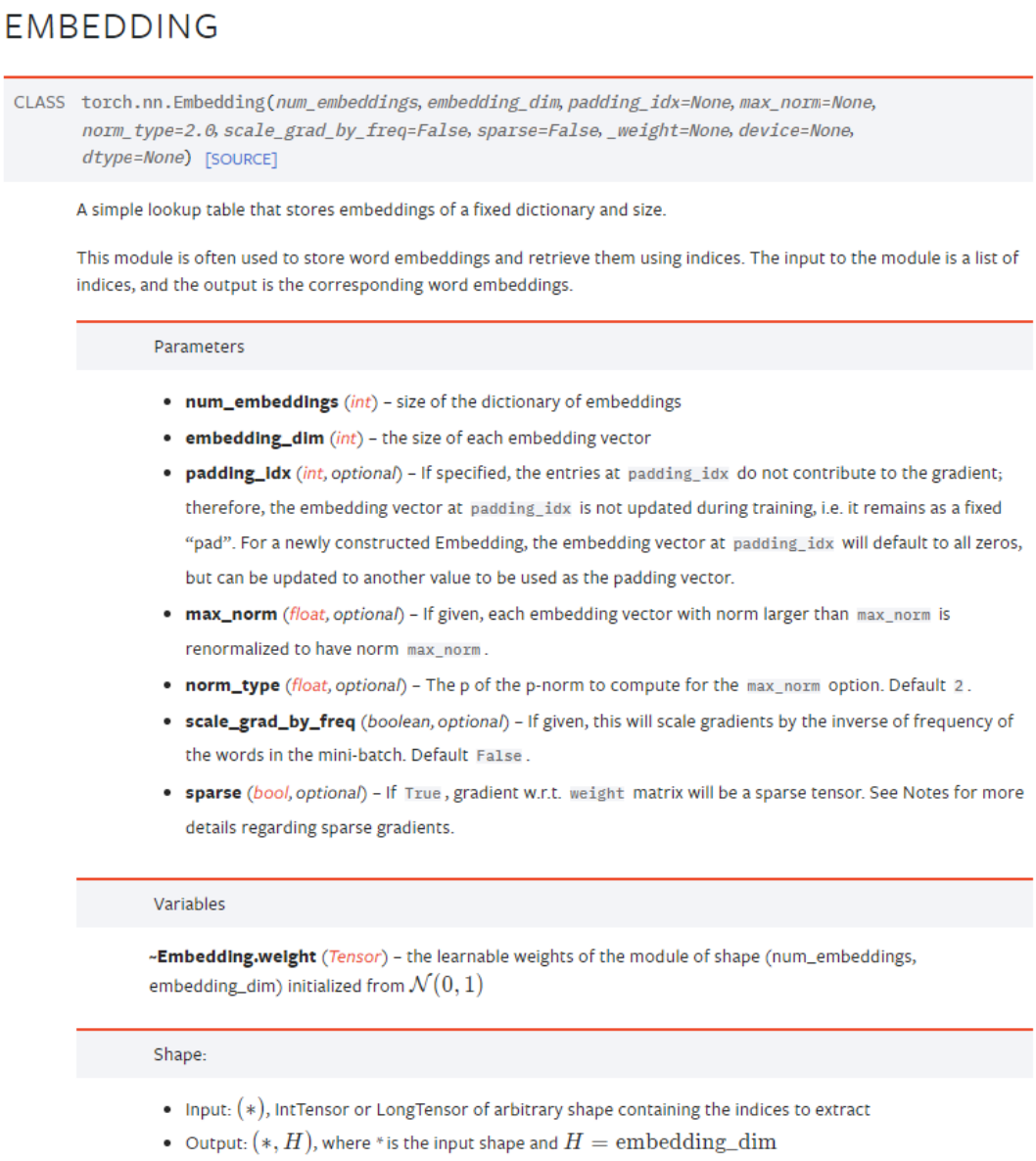


Figure 23. The diagram describing the methodology

4.2 Horse Attributes Embedding Scheme

In the first semester, every input was embedded as a text sentence, and every feature was converted as a word representation with the word embedding layer provided by PyTorch before entering the transformer encoder. This simulation of mimicking a race as a sentence and a feature as a word is retained but with a customized embedding scheme which produces better results.

4.2.1.1 Word Embedding Layer from PyTorch



EMBEDDING

CLASS torch.nn.Embedding(*num_embeddings*, *embedding_dim*, *padding_idx=None*, *max_norm=None*, *norm_type=2.0*, *scale_grad_by_freq=False*, *sparse=False*, *_weight=None*, *device=None*, *dtype=None*) [SOURCE]

A simple lookup table that stores embeddings of a fixed dictionary and size.

This module is often used to store word embeddings and retrieve them using indices. The input to the module is a list of indices, and the output is the corresponding word embeddings.

Parameters

- **num_embeddings** (*int*) – size of the dictionary of embeddings
- **embedding_dim** (*int*) – the size of each embedding vector
- **padding_idx** (*int, optional*) – If specified, the entries at `padding_idx` do not contribute to the gradient; therefore, the embedding vector at `padding_idx` is not updated during training, i.e. it remains as a fixed "pad". For a newly constructed Embedding, the embedding vector at `padding_idx` will default to all zeros, but can be updated to another value to be used as the padding vector.
- **max_norm** (*float, optional*) – If given, each embedding vector with norm larger than `max_norm` is renormalized to have norm `max_norm`.
- **norm_type** (*float, optional*) – The p of the p-norm to compute for the `max_norm` option. Default `2`.
- **scale_grad_by_freq** (*boolean, optional*) – If given, this will scale gradients by the inverse of frequency of the words in the mini-batch. Default `False`.
- **sparse** (*bool, optional*) – If `True`, gradient w.r.t. `weight` matrix will be a sparse tensor. See Notes for more details regarding sparse gradients.

Variables

~**Embedding.weight** (*Tensor*) – the learnable weights of the module of shape (num_embeddings, embedding_dim) initialized from $\mathcal{N}(0, 1)$

Shape:

- Input: $(*)$, IntTensor or LongTensor of arbitrary shape containing the indices to extract
- Output: $(*, H)$, where * is the input shape and $H = \text{embedding\_dim}$

Figure 24. The Interface of Word Embedding Layer from PyTorch [34]

In order to feed the input data into the word embedding layer from PyTorch [34], all features have to be first encoded into an integer value according to the ordinal encoding scheme. Every integer is treated as a word, and distinct integers represent different words. Suppose every input has n features and m is the expected dimension for each word, the input of the embedding layer is thus a sequence of integers with the length equal to n and the output of the layer is a matrix of size n x m so that m real values describe every word/feature. The details of the word embedding layer from PyTorch are shown in Figure 24.

4.2.1.2 Appropriateness of Word Embedding Layer for Horse Racing Data

From the perspective of the word embedding layer, it expects different integers represented distinct words from a bag of words. The information about the inequalities of numbers in the original data may lose after converting the data into sequences of words drawn from a bag of word because a bag of words does not characterize the relationship between numbers or words. For example, the original input contains the declared weights shown in Figure 25 with values 1098, 1103, 1331, etc. They may be encoded into integers of values 1, 2, 3 before feeding them into the embedding layer. We knew that 1331 is much greater than 1103 and 1098 is slightly smaller than 1103 but this relationship may not be well captured after embedding since 1 and 3 are expected to have equal distance from 2.

Also, the data contains the finishing time in types of real values, and it is unreasonable to encode real values into distinct integers and treats them as words. For example, it is not sensible to encode the real values 50.5, 51.0, 55.5, …. into integers

1, 2, 3, …. as the distance relationship is not preserved, and every real value must occupy as a unique word which hugely increases the size of the bag of words. Therefore, the word embedding layer is not appropriate for horse racing data.

horse_1_jockey_name	horse_1_trainer_name	horse_1_actual_weight	horse_1_declared_weight	horse_1_finish_time	h
DWhyte	JSize	133	1098	01:09.9	
CWWong	CHYip	128	1103	01:36.3	
DWhyte	ALee	129	1331	01:10.0	
GBoss	DEFerraris	131	1107	01:25.1	
BPrebble	DJHall	133	1071	01:09.9	
KTYeung	ATMillard	128	1032	01:23.5	
MNunes	PO'Sullivan	133	1117	01:10.2	
GBoss	JMoore	128	1034	01:08.9	
ZPurton	PFYiu	133	1136	01:09.4	
FCoetzee	JSize	132	1117	01:36.8	
CWWong	CHYip	129	1151	01:27.1	
ZPurton	ASchutz	133	972	00:58.8	
ESaint-Martin	ASchutz	133	1035	01:42.6	
KLChui	YSTsui	128	1140	01:09.8	
YTCheng	TWLeung	131	960	02:08.1	
ZPurton	ASchutz	133	1103	01:30.8	
DBeadman	SWoods	132	1200	01:42.0	
ZPurton	DEFerraris	132	1152	01:09.9	
CWWong	CHYip	128	1098	02:10.8	
MDuPlessis	PFYiu	129	1125	01:13.1	
YTCheng	KWLui	129	1175	01:27.3	
GMosse	ALee	133	1024	01:25.0	
HWLai	LHo	131	1044	01:11.2	
CWWong	DCruz	128	1151	01:23.7	
MWLeung	KWLui	130	1141	01:37.4	
ODoleuze	CFownes	133	1131	01:11.0	
KLChui	PO'Sullivan	128	1127	01:50.3	
GBoss	ALee	131	1198	01:23.7	
ESaint-Martin	CSShum	133	1033	01:22.8	

Figure 25. A Subset of Input Features

4.2.2.1 Word Embedding Simulation by Horse Token

As the word embedding layer is inappropriate for horse racing data, the embedding scheme is customized to suit the data. The input of the neural network was initially been a sequence of words, but it is now a sequence of horse tokens. The main idea of horse token generation is that some real values describe both horse tokens and words after embedding. A comparison between word embedding and horse token generation is provided as follows.

```
>>> # an Embedding module containing 10 tensors of size 3
>>> embedding = nn.Embedding(10, 3)
>>> # a batch of 2 samples of 4 indices each
>>> input = torch.LongTensor([[1,2,4,5],[4,3,2,9]])
>>> embedding(input)
tensor([[[-0.0251, -1.6902,  0.7172],
         [-0.6431,  0.0748,  0.6969],
         [ 1.4970,  1.3448, -0.9685],
         [-0.3677, -2.7265, -0.1685]],

        [[ 1.4970,  1.3448, -0.9685],
         [ 0.4362, -0.4004,  0.9400],
         [-0.6431,  0.0748,  0.6969],
         [ 0.9124, -2.3616,  1.1151]]])
```

Figure 26. Example of Word Embedding [34]

Suppose a sequence of 4 words encoded as a sequence of integers 1 ,2 ,4, 5 is embedded with the word embedding layer and the embedding dimension is 3 as shown in Figure 26. Then, the output of the embedding layer is a matrix of size 4 x 3 such that each row represents a word, each column represents a dimension of the words, and each element is a real value.

```
[
  [ x.xxx, x.xxx, x.xxx ],    (horse token 1)
  [ x.xxx, x.xxx, x.xxx ],    (horse token 2)
  [ x.xxx, x.xxx, x.xxx ],    (horse token 3)
  [ x.xxx, x.xxx, x.xxx ],    (horse token 4)
]
```

Figure 27. Example of Horse Tokens

Similarly, suppose there is a horse race with only 4 horses, and we want the output of our customized embedding to be a sequence of 4 horse tokens. Each token has 3 dimensions so that 3 real values describe every horse token as the matrix shown in Figure 27.

4.2.2.2 Horse Tokens Generation by PCA

In the generation of horse tokens, the horse racing dataset consisting of 9191 races is partitioned into 14 matrices. The matrix i contains only attributes of the horse with number i in all 9191 races and the size of the matrix was 9191 x n where n is the number of attributes. Principal component analysis (PCA) is utilized to reduce the dimensionality of every matrix from n to m. The generation process is illustrated in Figure 28.

All 14 matrices are concatenated horizontally after PCA so that a matrix of size 9191 x 14 x m is obtained. The first dimension is the index for the race, the second dimension is the index for the horse token and the third dimension is the index for real values which describe the token.

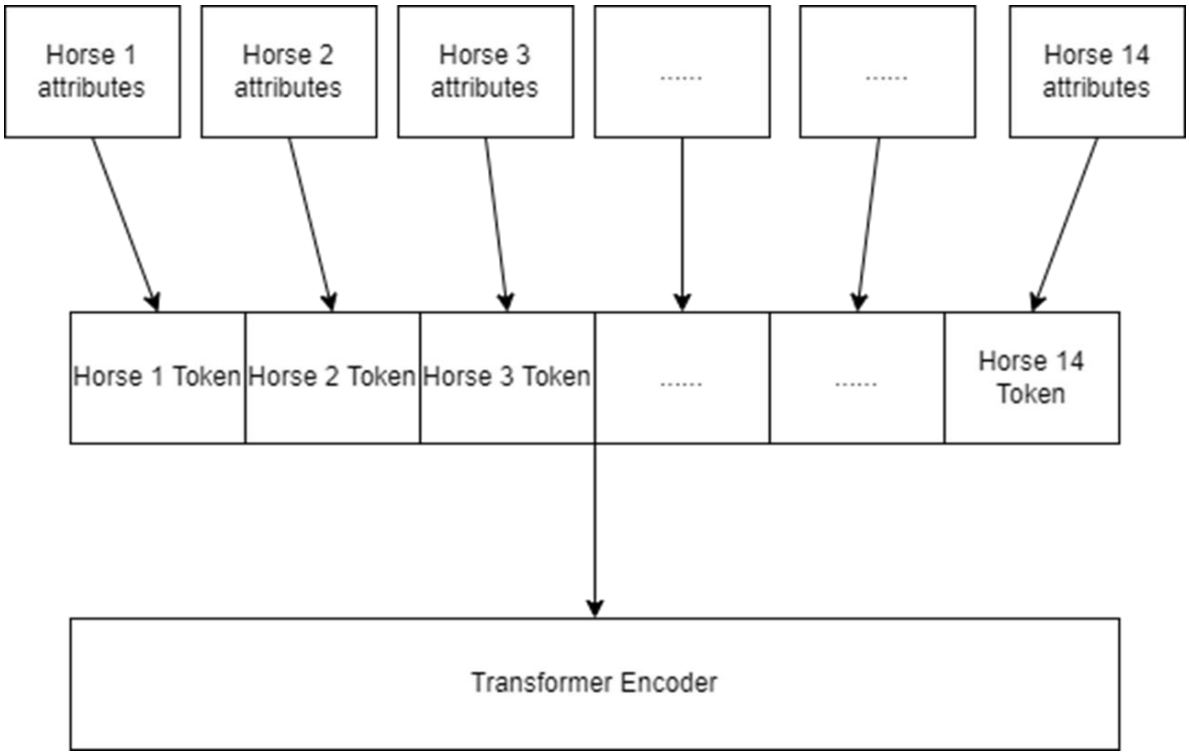


Figure 28. Horse Tokens Generation by PCA

4.3 Model Design

4.3.1 Multilayer Perceptron Classification

The number of classes in our multilayer perceptron equals the number of horses in the race. For instance, there will be 14 classes if the race has 14 participating horses, and each class corresponds to a horse number. The input is a race record joined with horse records according to the horse names listed in the race record. The neural network's output is a vector consisting of the values resembling the probabilities of winning horses, and classification is done based on these values. If the horse with horse number 7 wins the race and our multilayer perceptron predicts it correctly by giving it the highest value in the vector, the model assigns this input to class 7.

The multilayer perceptron has a total of 5 linear layers. The first linear layer is the input layer that takes the input vector's values. There are 3 hidden linear layers with a number of neurons in the range of 100 – 400 to increase the model's sensitivity in the learning process [35]. The output of each hidden linear layer has to be passed through the ReLU activation function, which determines the activity of the neurons. For the second and third hidden layers, dropout layers are inserted for regularization, which helps the model avoid overfitting by randomly losing connections between neurons in the training process [36]. The last linear layer is the output layer storing the outcome. We pick the cross-entropy function and stochastic gradient descent as the model's loss function and optimizer. The diagram of the model is shown in Figure 29.

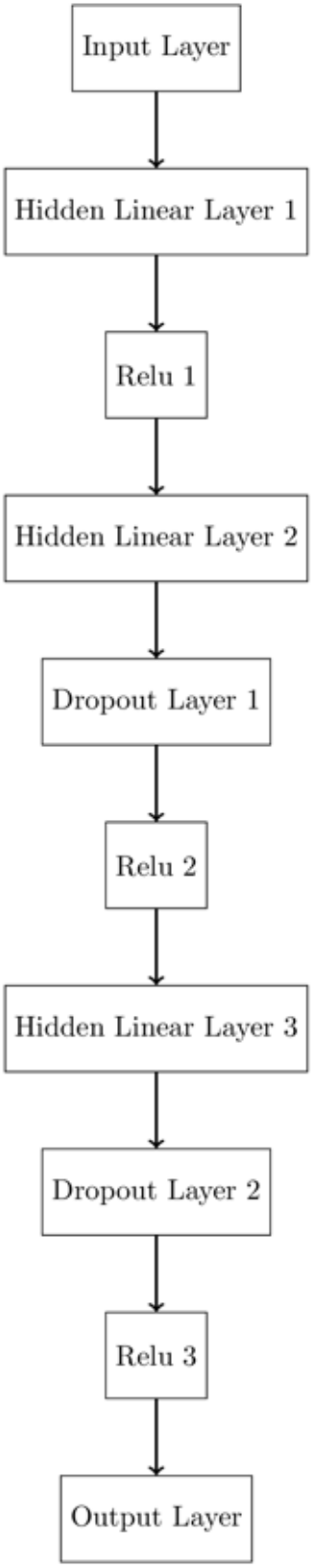


Figure 29. The Multilayer perceptron architecture

4.3.2 Transformer Classification

As our ultimate goal of designing two neural network models is to compare the effect of using different neural network architectures in horse racing prediction, we decide to have a similar setting regarding the input and output in the transformer model as the multilayer perceptron. Therefore, the input of the transformer classification is joined by race records and horse records, and the output is the horse number belonging to the winning horse.

The transformer model does not use the decoder [25] mentioned in the original paper because the output of our classification problem is a single number instead of a sequence. We partition our model into three stages. The first stage is about the data formatting of the input vector. We use an embedding layer to increase the dimension of each feature which mimics the word embedding in natural language processing. Then, we use a position embedding layer to remember the position of each feature as the position is meaningful in our input data which features of one horse are in closer distance than other features. Next, the processed input enters the encoder of a transformer to learn the dependencies between features. The output of the encoder is sent to a simple, fully connected feedforward network consisting of 2 hidden linear layers and an output layer. We pick the cross-entropy function and stochastic gradient descent as the model's loss function and optimizer. The diagram of the model is shown in Figure 30.

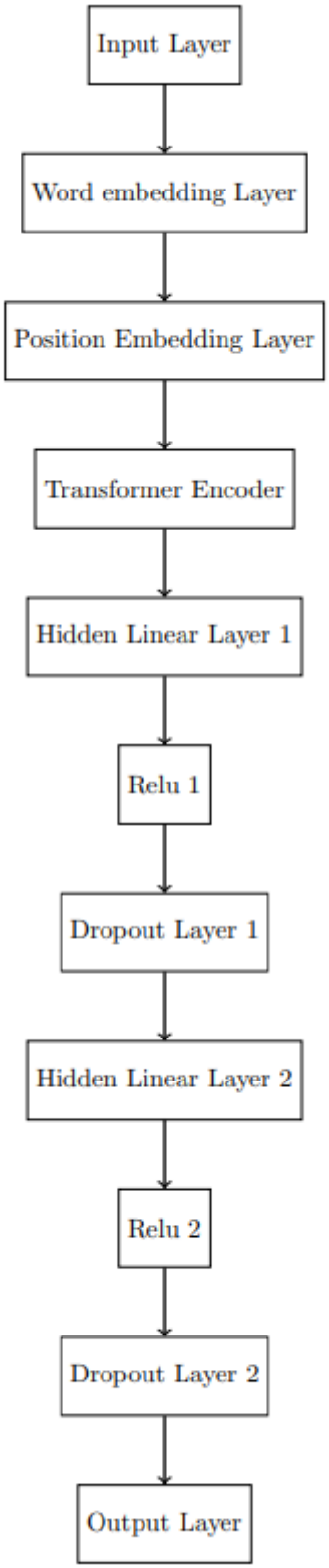


Figure 30. The Transformer architecture

Chapter 5

Experiments and Results

Our experiment has two phases. In the first phase, we have trained three models with changes in input features and the neural network architecture. The first model is the multilayer perceptron classification model with inputs including ratings. The second model is the transformer classification model with inputs excluding ratings. The third model is the transformer classification model with inputs including the rating. We want to study whether the third model achieves a better result. Therefore we use the results of the first and the second models to be the reference when evaluating the performance of the third model. After the model selection, we figure out the more appropriate embedding scheme among the word embedding and horse token embedding through their performance on the best model found in the first stage.

5.1 Input Data

We separate the most recent 688 horse races between 9 December 2020 and 10 October 2021 from our original horse race dataset for testing. The remaining 8503 horse races are used in the training process. Splitting the training data and testing data randomly is inappropriate in our context because we are more interested in correct predictions of new races, and the past races having retired horses should not be involved in the test data when we want to evaluate the performance of our models in predicting the new races.

We formulate each race as a single input after data preprocessing, as shown in Table 5. All information about a race, including the track's conditions, attributes of horses,

and ratings, are packed into a row in our input matrix. This ensures that the neural network receives sufficient data when predicting the winning horse in a race. For further comparison of different combinations of neural network architectures and data, a few columns in the input matrix are discarded to study the effect of the discarded features.

Feature	Description
Venue	Location of the race
Horse_class	Class of the horses Stronger horses compete in high race class
Distance	The distance of the race
Going	Condition of the lane
Course_track	The lane of the race
Course_track_code	Description about the lane
Horse_i_number	The horse number in the race
Horse_i_name	The name of horse
Horse_i_jockey	The name of jockey
Horse_i_trainer	The name of trainer
Horse_i_declared_weight	The weight of horse
Horse_i_origin	The place of birth
Horse_i_age	The age of horse
Horse_i_color	The color of skin
Horse_i_sex	The gender of horse
Horse_i_1st_place_frequency	The frequency of getting 1st place
Horse_i_total_race	The total count of horse's participation
Horse_i_rating	The rating of the horse

Table 5. The schema of the input data

5.2 Model Comparisons

This section evaluated several combinations of models and rating systems to find the best model and ratings input in the horse racing context. The performance evaluation was based on the strategies proposed in chapter 2.3, which mainly focused on the test accuracy and profits in betting simulation.

5.2.1 Multilayer Perceptron

5.2.1.1 Accuracy

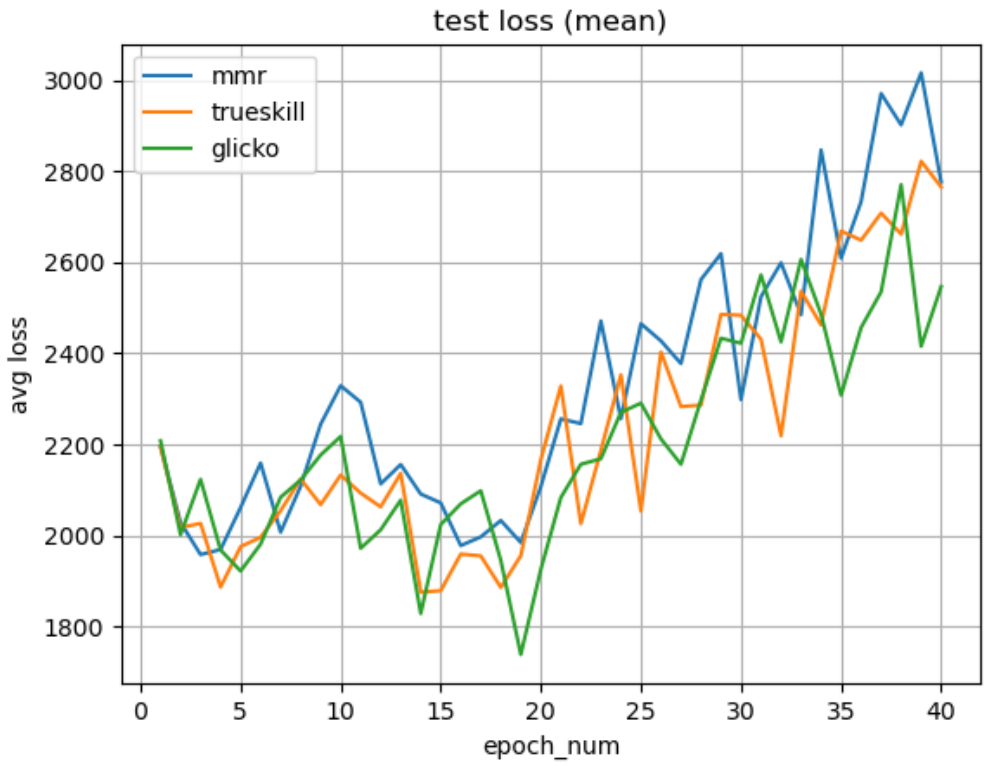


Figure 31. The loss of multilayer perceptron on test data

In Figure 31, this graph shows the average loss of all batches on the test data with respect to the training epoch number. Three curves represent the average loss of the models with different ratings in the input. In the graph, we see that the average loss of

all three models has a general decreasing trend from epoch number 1 to epoch number 17. The multilayer perceptron with Elo-MMR rating input has a low average loss at epoch number 18. The multilayer perceptron with Glicko ratings as input has a low average loss at epoch 18. The multilayer perceptron with the TrueSkill rating as input has a low average loss at epoch 17. After epoch number 18, the average loss of all three models increases remarkably, indicating the overfitting. The model with the Elo-MMR rating as input has the highest average loss among the other models, while the model with the Glicko rating as input has the lowest average loss among the other models.

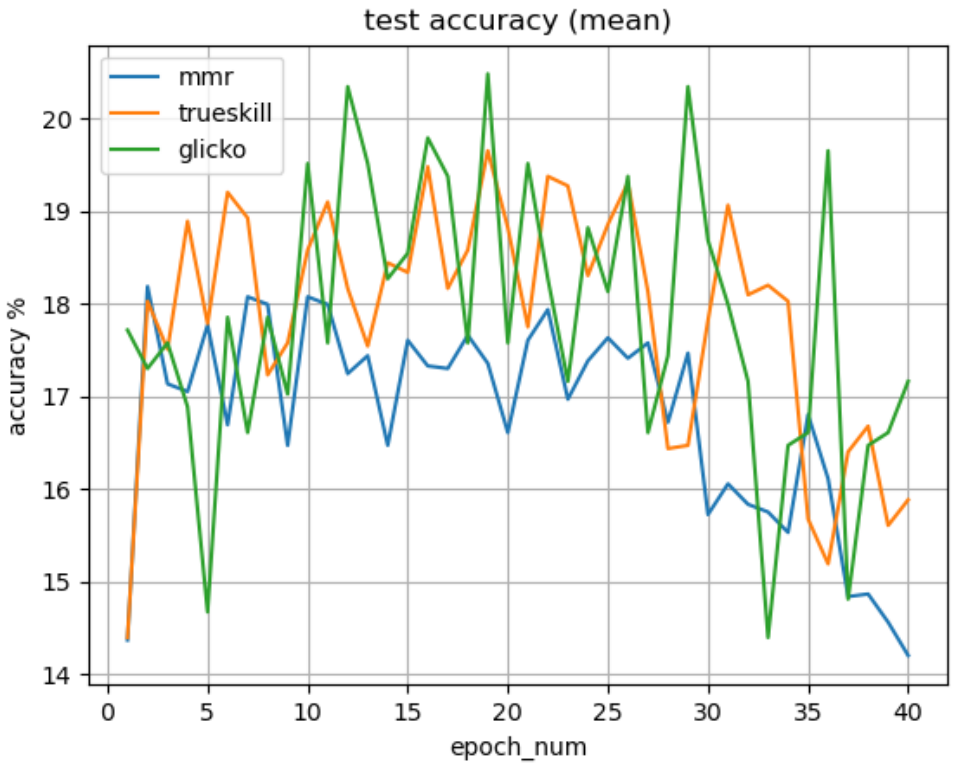


Figure 32. The accuracy of multilayer perceptron on test data

We observe the testing accuracy in Figure 32. We notice that the testing accuracy of all three models keeps dropping after the epoch number 17. This is because the models overfit, as reflected in Figure 32. The model with the Glicko rating reaches the

highest test accuracy of 20.4%, while the model with the Elo-MMR rating has the lowest accuracy among the other models. This is related to the same pattern in the graph of average loss in Figure 32. We can also see that the model's accuracy with Glicko fluctuates in a larger range than that with Elo-MMR and TrueSkill because the Glicko rating is dedicated to 2 player games while Elo-MMR and TrueSkil ratings are dedicated to multiplayer games.

5.2.1.2 Betting Simulation

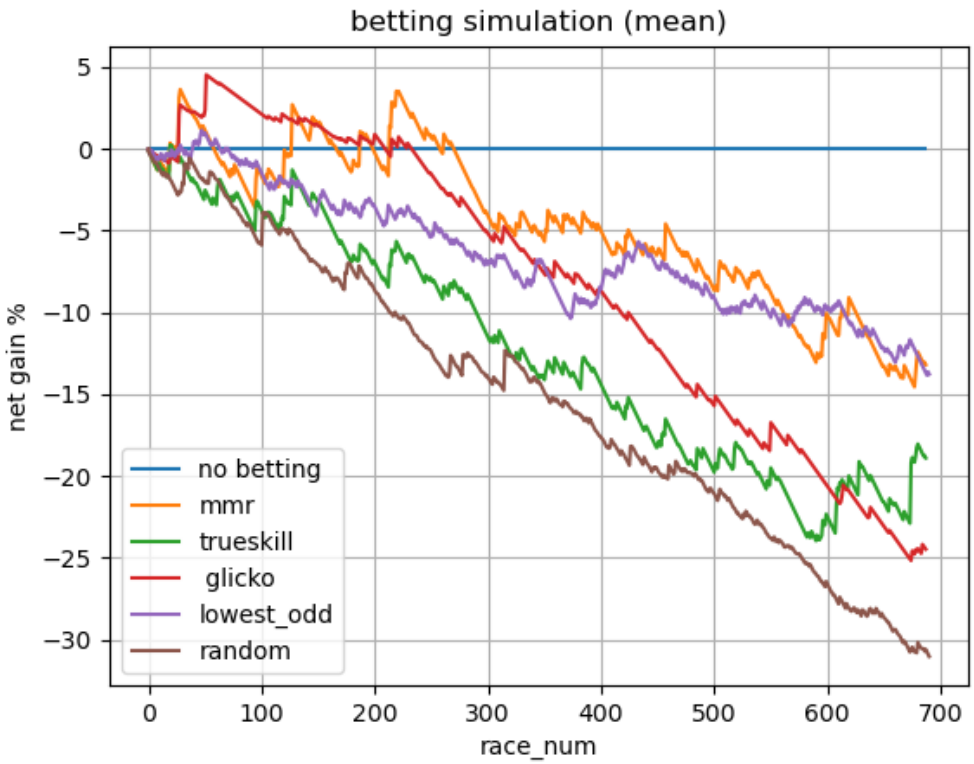


Figure 33. The betting simulation of multilayer perceptron on test data

We use the prediction of the betting models and show the result in Figure 33. In our betting simulation, all three models perform better than random betting. The model with the Elo-MMR rating has a similar performance as the lowest odd betting, reflecting public intelligence. This means that the Elo-MMR rating is comparable to

the win odds in betting guidance. However, none of the models can give us a positive net gain.

5.2.2 Transformer classification without Ratings

5.2.2.1 Accuracy

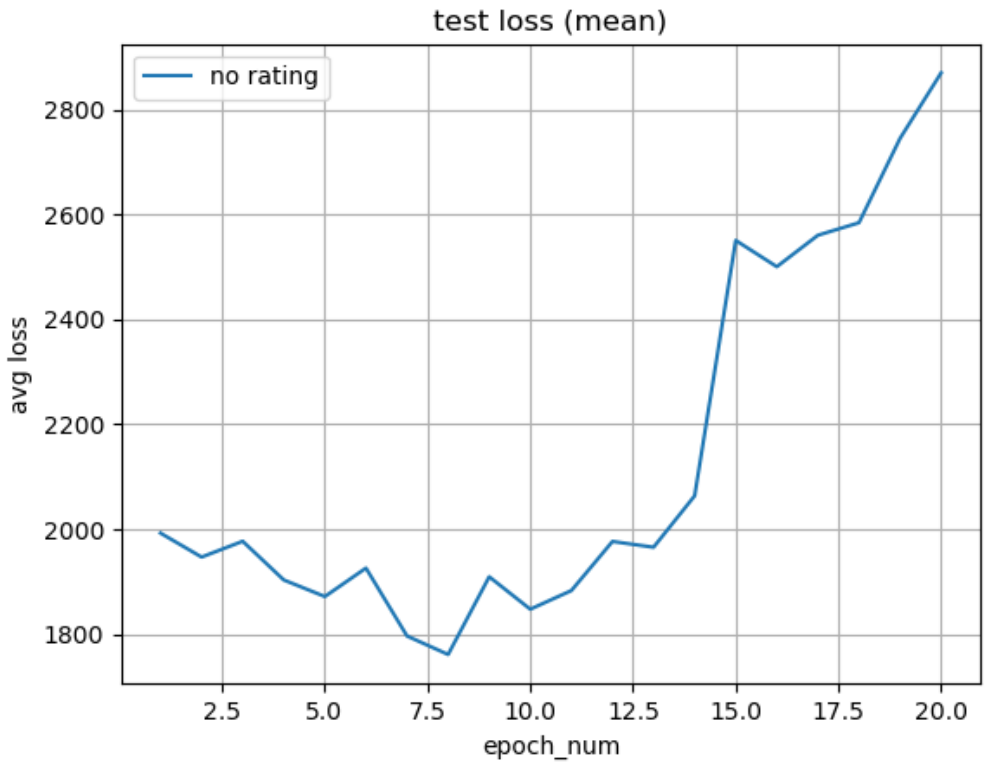


Figure 34. The loss of transformer on test data without rating

In Figure 34, this graph shows the average loss of all batches on the test data with respect to the training epoch number. In the chart, we see that the average loss of this model has a general decreasing trend from the start to epoch number 8. After epoch number 8, the average loss of the model increases remarkably, which indicates the overfitting. We observe that this model reaches the converges earlier than multilayer perceptron models. The transformer classification model is more complex than the multilayer perceptron and it learns faster.

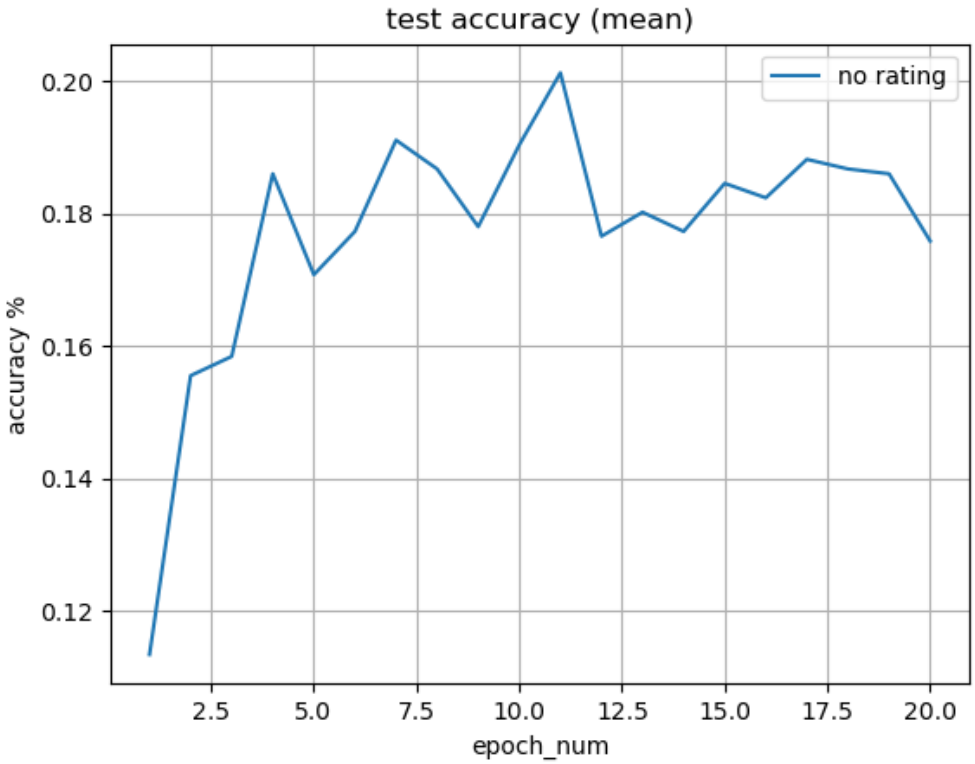


Figure 35. The accuracy of transformer on test data without rating

We examine the testing accuracy in Figure 35. We see that the testing accuracy of this model is in the range of 17% to 20% for epoch numbers larger than 3. The reason for considering the test accuracy after epoch number 3 is that the model is learning, and its average loss on test data has not reached the minimum before epoch number 3. From the implication of the average loss in Figure 34, the best performance of this model has 19.2% at epoch number 6.

5.2.2.2 Betting Simulation

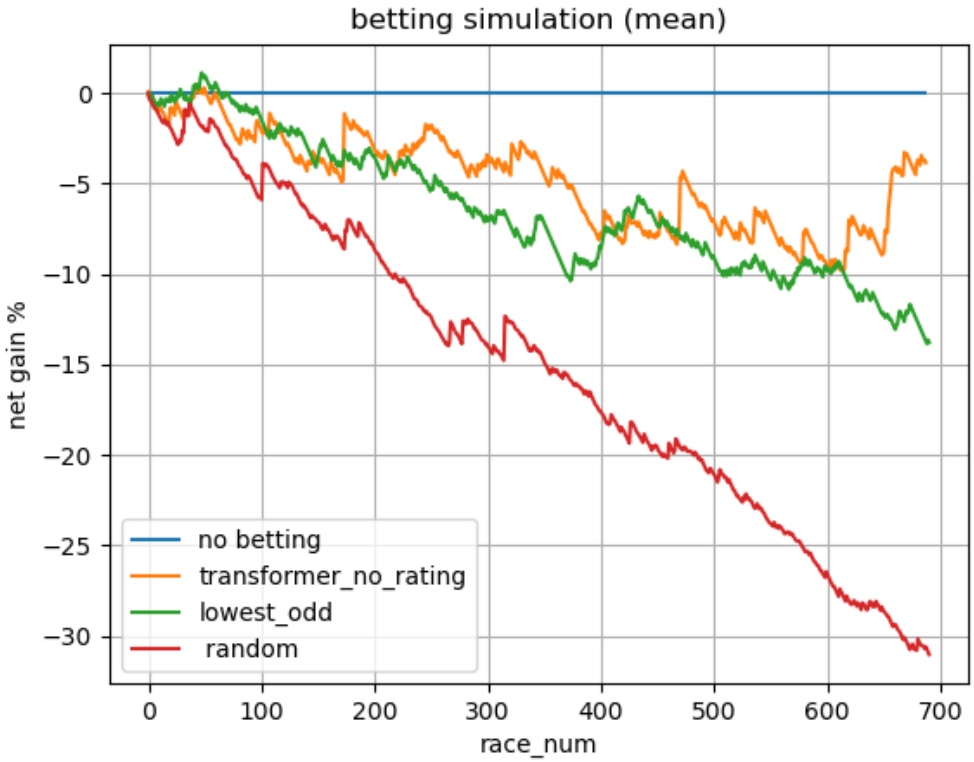


Figure 36. The betting simulation of transformer on test data without rating

We use the prediction of the transformer model to guide our bet on the test data. Figure 36 reveals the performance of this model in the profit-making aspect. Our test data shows that the net gain is -4% after betting on all 688 races. The performance of this model in betting is better than the multilayer perceptron, which has -13% as the highest net gain with the Elo-MMR rating included in the input.

5.2.3 Transformer classification with Ratings

5.2.3.1 Accuracy

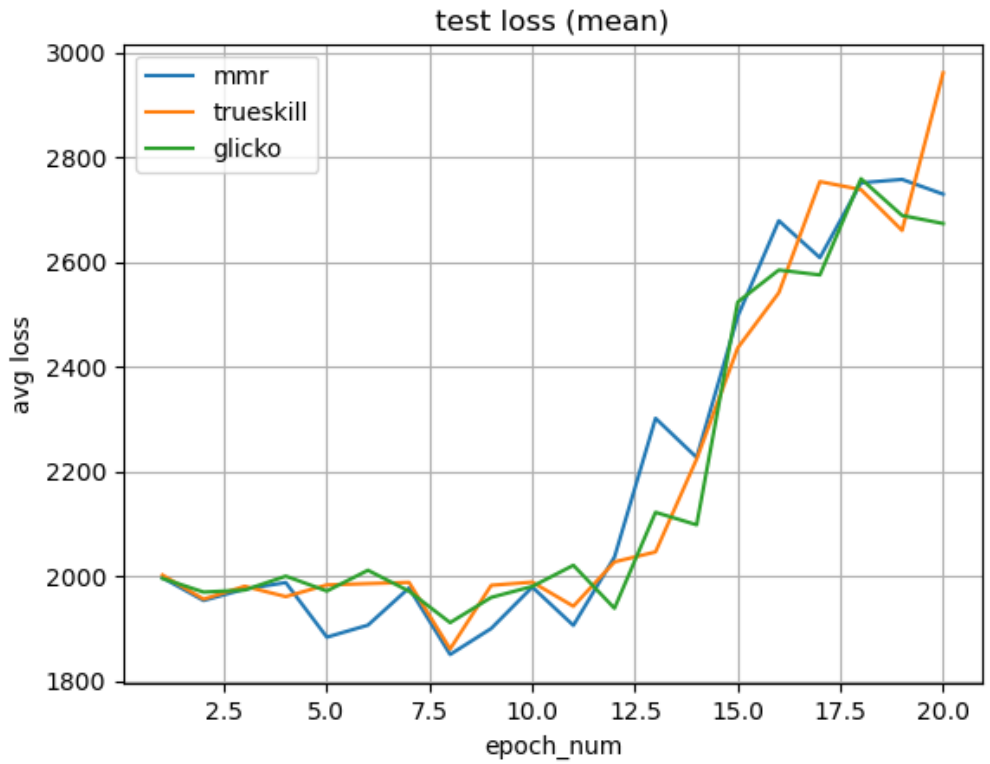


Figure 29. The loss of transformer on test data with rating

Figure 37 shows the average loss of transformer models with different ratings involved in the input. All models overfit after the epoch number 8 because their average loss on test data keeps increasing after the epoch number 8. When compared to the graph for models using multilayer perceptron, we see that using different ratings is not significant here because the differences in average loss between the transformer models that use different ratings are more minor than that of the multilayer perceptron models.

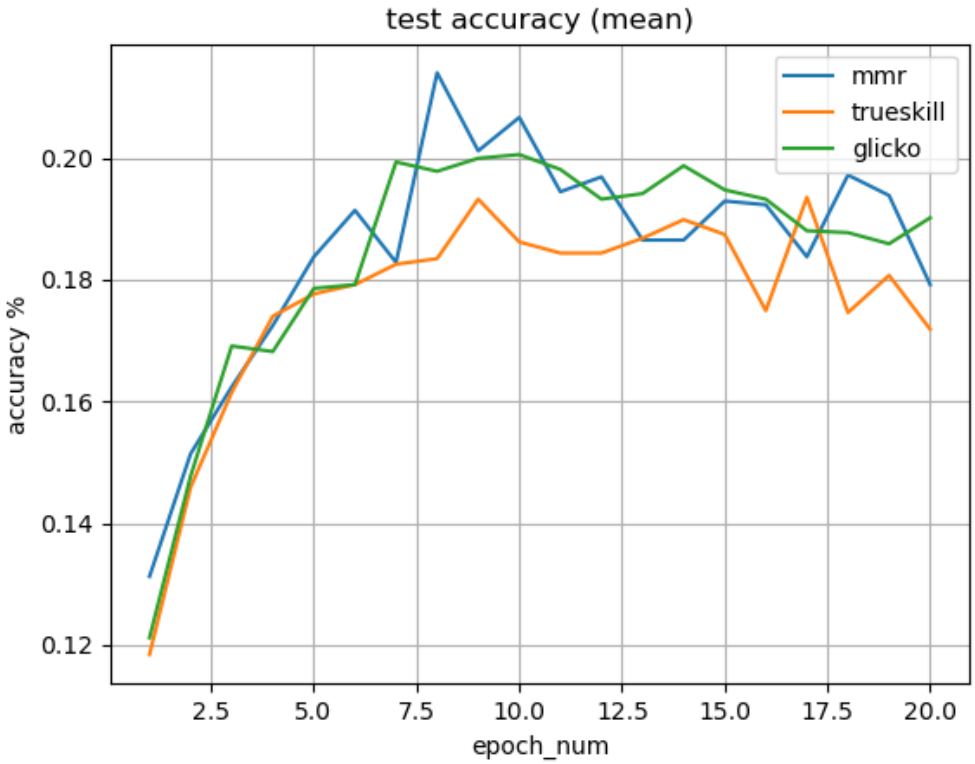


Figure 38. The accuracy of transformer on test data with rating

As Figure 37 indicates overfitting after epoch number 8, we focus on the test accuracy before epoch number 8. We notice that the test accuracy increases consistently from the start. The transformer model with the Elo-MMR rating included has the highest test accuracy of 21.4% among the other models. Compared to the test accuracy of the transformer model without rating in the input, we conclude that including ratings in the transformer model as input slightly increases the test accuracy. Compared to the test accuracy of multilayer perceptron models, we conclude that using the transformer model slightly increases the test accuracy and narrows down its confidence level because its fluctuation is slight, as shown in Figure 38.

5.2.3.2 Betting Simulation

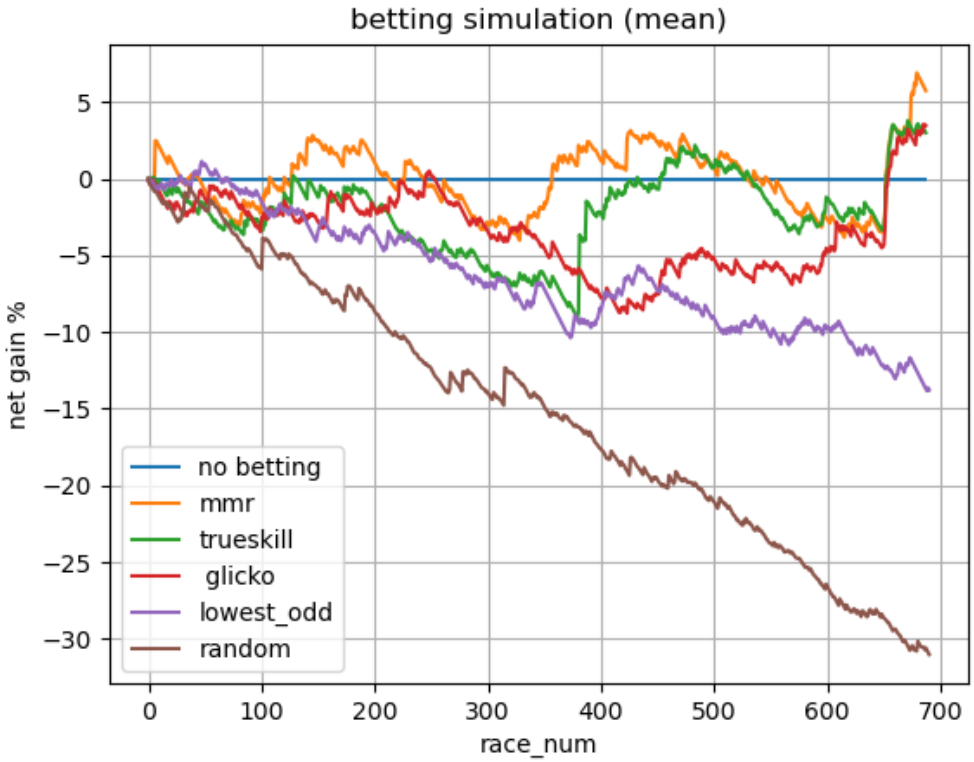


Figure 39. The betting simulation of transformer on test data with rating

When following the predictions of these transformer models in betting, we obtain a satisfactory result. The models give us a positive net gain of 3% to 6% after betting on 688 races in the test data, as shown in Figure 39. The transformer models with rating do have a better performance than the transformer model without rating and the multilayer perceptron models with rating. Also, we find that the change of net gain is confined to a smaller interval throughout the betting simulation when using the transformer model with the Elo-MMR rating.

5.3 Embedding Methods Comparison

Beyond the neural network architecture and rating estimation system, input embedding is an essential ingredient in the learning process. A better embedding method could provide a more precise vector representation of input carrying semantic meanings. Therefore, a proper embedding method helps the model better understand the similarities and differences of inputs and affect the model's performance. During the investigation, the inappropriateness of fitting horse racing input into word embedding is spotted, and the horse token embedding is then advocated to replace the word embedding. So, the two embedding methods are compared under the same architecture and rating input, the transformer model with Elo-MMR ratings, to determine a more suitable embedding method. As the result of using word embedding in the transformer model with Elo-MMR ratings is shown in Figure 38, we only offer the result of using horse token embedding with the specified transformer below.

5.3.1 Transformer with Horse Token Embedding

5.3.1.1 Accuracy

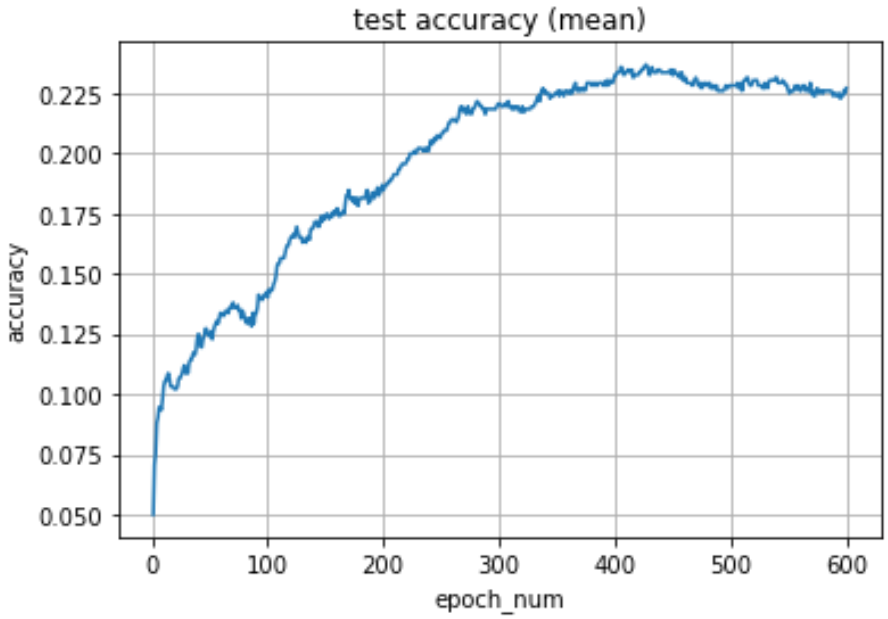


Figure 40. Test accuracy when horse token embedding was used

Overfitting occurs when the transformer model, which uses word token embedding, is applied to the input with horse token embedding. Therefore, the number of encoder layers is decreased to form a simpler transformer model, but more epochs are needed instead. The maximum accuracy of 23.4% was reached with 420 epochs, as shown in Figure 40. Compared to the accuracy of using the word embedding layer, the accuracy of using the horse token embedding is increased by 2%, from 21.4% to 23.4%. The accuracy is stabilized at around 22.5% in later epochs after reaching the maximum, and it differs from the performance shown in Figure 37 that the accuracy has a significant drop of 2% after its maximum. The difference evinces that the horse token embedding is more applicable for input in the horse racing context.

5.3.1.2 Betting Simulation

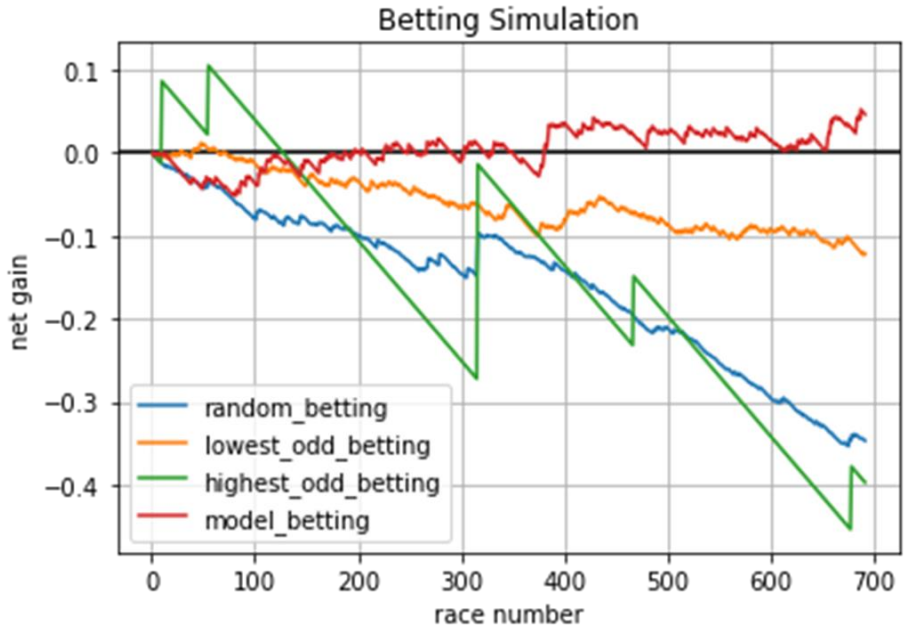


Figure 41. The betting simulation of transformer with horse token embedding

The replacement of the word embedding by the horse token embedding excels in the profit-making capability of our transformer model. From Figure 41, the net gain over the races in test cases has a gentle trend of increase in which the profits from the correct predictions compensate for the losses caused by the wrong predictions. The steady growth of the net gain implies that the horse token embedding is more favorable than the word embedding, which leads to large fluctuations in the net profit, as shown in Figure 39, towards the profit-making aspect.

Chapter 6

Interpretability

6.1 Assessment to Model Capability

The comparison between a transformer classification model and a multilayer classification model discussed in Chapter 5 exhibits the privilege of the transformer classification model in the horse racing context. Also, horse token embedding instead of word embedding leads to higher accuracy in winning horse prediction. These observations bring interest in investigating how information encoded in horse token embedding allows the transformer model to perform better. One possible way to examine the information, the internal vector representation, carried in every layer of the encoder layer in the transformer is using probing tasks for reasoning the capabilities of the transformer in relation to properties of a race that helps in correct prediction [37].

Regarding the process of learning useful properties for a correct prediction, we lay down an assumption here, and we will revisit them with the results obtained after the probing tasks. We assume that layers at the front of the encoder learn only simple features and layers at the end of the encoder learn more abstract concepts. The internal vector representations at the layers towards the end of the encoder should have more helpful information about the properties of a race.

6.1.1 Probing Model

The probing model could partially reveal the information captured in the internal vector representations because the probing model could have accurate predictions if the information in the internal vector representation is sufficient for the probing model to make correct decisions. The precise prediction of the probing task for a particular property entails the capability of the transformer model to learn that property because it has sufficient information stored in the interval vector representation for a better understanding of that property [38].

A probing dataset is constructed for each property that will be investigated. In a probing dataset, the input of the probing model is the internal vector representation, while the target is an integer indicating the class of the input. The architecture of the probing model is illustrated in Figure 42. The internal vector representation in the probing dataset is extracted from the transformer model, which will be assessed and fed to a simple multilayer perceptron for binary classification. To avoid overfitting, the multilayer perceptron consists of two fully connected layers with ReLU activation and dropout. The output of the multilayer perceptron is then compared to the corresponding target in the dataset for evaluation. This probing model could be generalized for multi-class classification by allowing the multilayer perceptron to have more than two outputs if the property has more than two expressions.

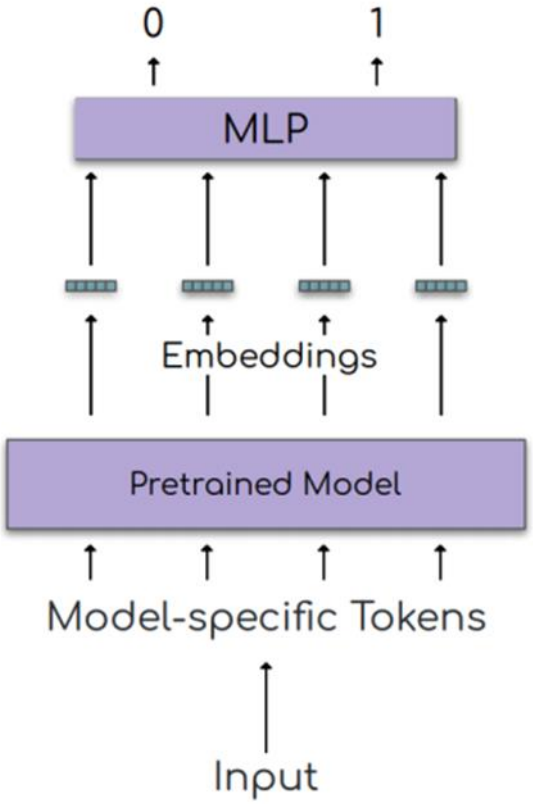


Figure 42. The probing model [38]

6.1.2 Number of Participants

The number of participants varies in races, and it is usually in the range of 10 to 14. While maintaining the input to have 14 horse tokens consistently even though the number of participants is sometimes fewer than 14, the model is expected to know the number of the participant when predicting the winner horse so that the prediction of the winning horse number is within the range. For example, the transformer's output should be in the range of 1 to 10 if the number of participants is 10. Therefore, the capability of understanding the number of participants is investigated to see if the model can avoid giving an out-of-bound prediction to minimize the probability of producing an unreasonable forecast.

In the generation of the probing dataset, the count of participants in each race is extracted from the data frame of the original dataset, and it is marked as the target for that particular race. The input x_i for each race is a row vector of length n where n is the length of the internal vector representation and i was the i-th layer in the transformer encoder for $1 \leq i < 5$. The target y for each race is an integer such that $1 \leq y \leq 5$ represents 5 possible numbers of participants. The probing task is thus a classification task of 5 classes.

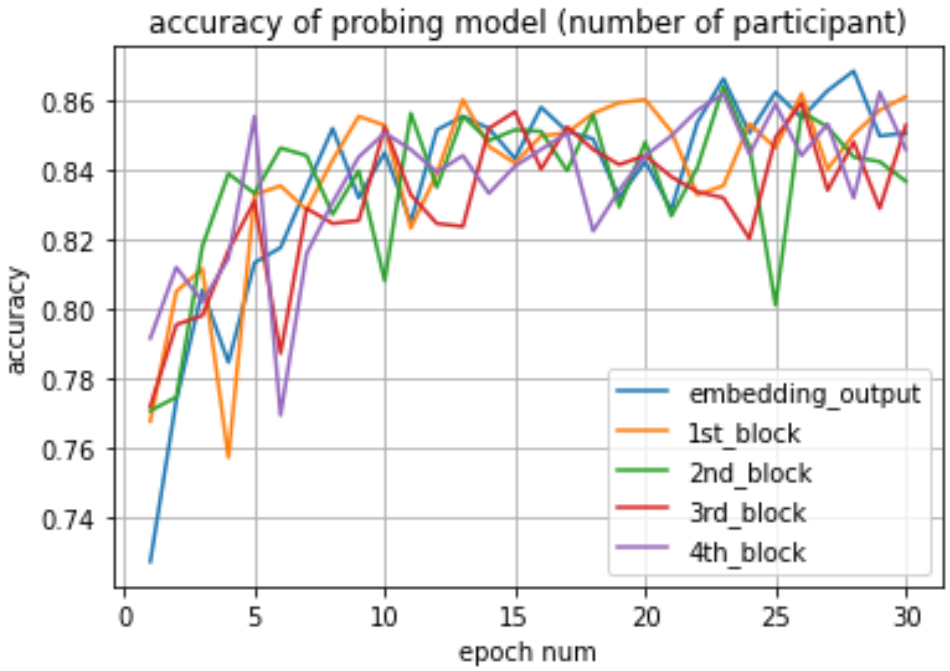


Figure 43. Accuracy of probing model (number of participants)

The accuracy of classifying the number of participants in all layers is at least 86%, as shown in Figure 43. The high accuracy indicates the capability of the transformer to identify the number of participants correctly in most cases. This essentially reduces the probability of incorrect prediction due to misunderstanding of race conditions by the model and pushes the forecast towards the ground truth. Another substantial interpretation is the preservation of information about participant count over all layers, as we see those similar accuracies in determining the participant count in all

layers. One possible explanation is that the knowledge of participants is essential in predicting the winning horse so that the model does not forget it but preserves it.

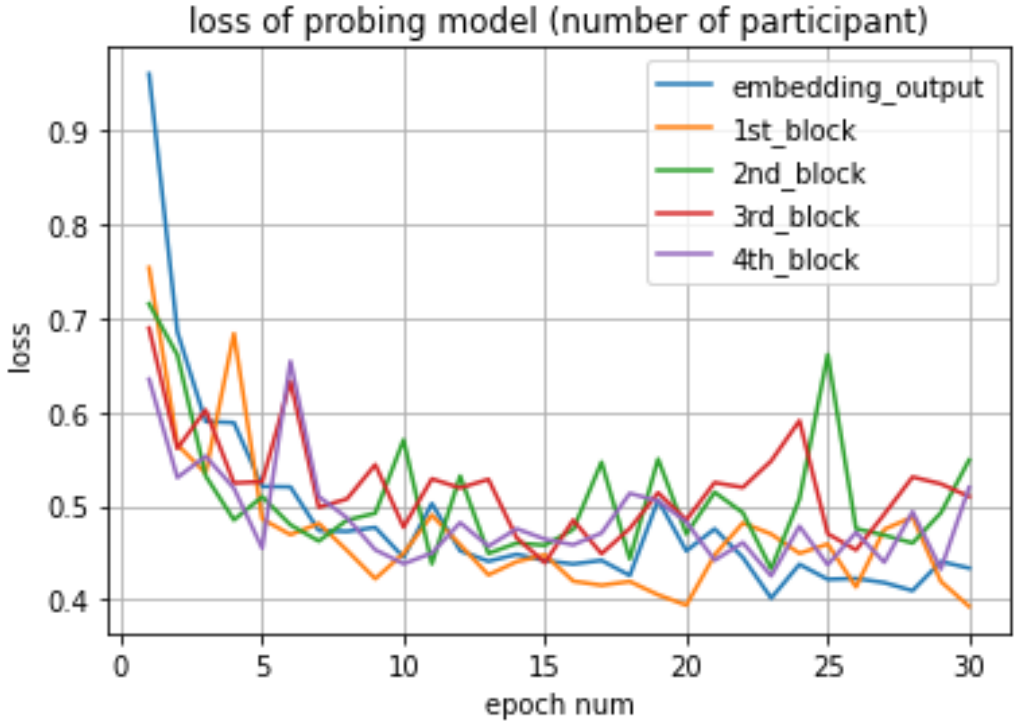


Figure 44. Loss of probing model (number of participants)

We expected more complex concepts are learned by later layers, like the aggregation of the existence of horses into count, while earlier layers learn only simple features such as the dichotomy in regard to the existence of a particular horse. Although the result shown in Figure 44 did not strongly verify the assumption, it does not disagree at all as the loss of model using the fourth encoder layer is lower than that of the second and third layers. Therefore, further investigation into other properties is needed to verify our assumption.

6.1.3 Most Popular Horse

Wining odds could improve the model's performance since it involves the public intelligence and reflects the most popular horse with the lowest win odds. However, it is excluded from our input because it keeps changing until the start of the race and we want to resemble the effect of win odds by static rating and transformer architecture. To determine whether rating and transformer could have a similar impact on finding the most popular horse, we study whether the internal vector representation contains enough information for identifying the most popular horse.

The input in the probing dataset was the same as the input in section 6.1.2. The target of the dataset is the horse with the lowest win odds as the win odds of the horse decrease if more people bet on that horse. So, target y will be an integer such that $1 \leq y \leq 14$ indicates the horse number of the winner. The probing task is thus a classification task of 14 classes.

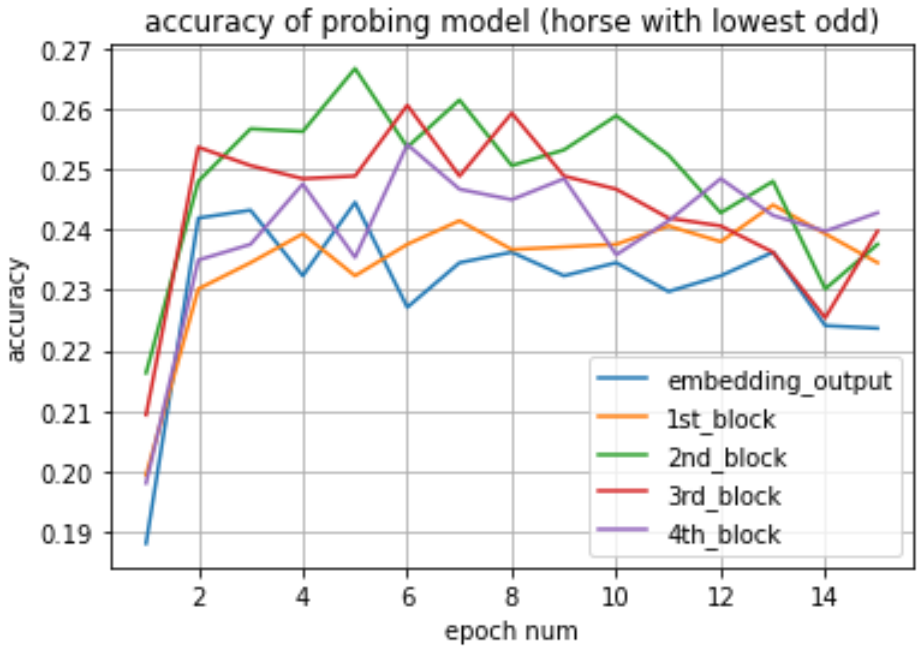


Figure 45. Accuracy of probing model (most popular horse)

In Figure 45, the model using vector representation of the second encoder layer as input reached the maximum accuracy of 26.6%, followed by that of the third encoder layer, the fourth encoder layer, the embedding layer, and the first encoder layer, which had maximum accuracies of 26.1%, 25.4%, 24.4%, and 24.2% respectively. The transformer indeed learned to distinguish the most popular horse, albeit with a seemingly low accuracy of 26.6%. There were 14 horses in a race, and the probability of selecting the most popular horse in a random guess was 0.0714, but the accuracy was boosted to 0.266 when the internal vector representation was used to assist the selection. The model could find the most popular horse indicated by the lowest win odds.

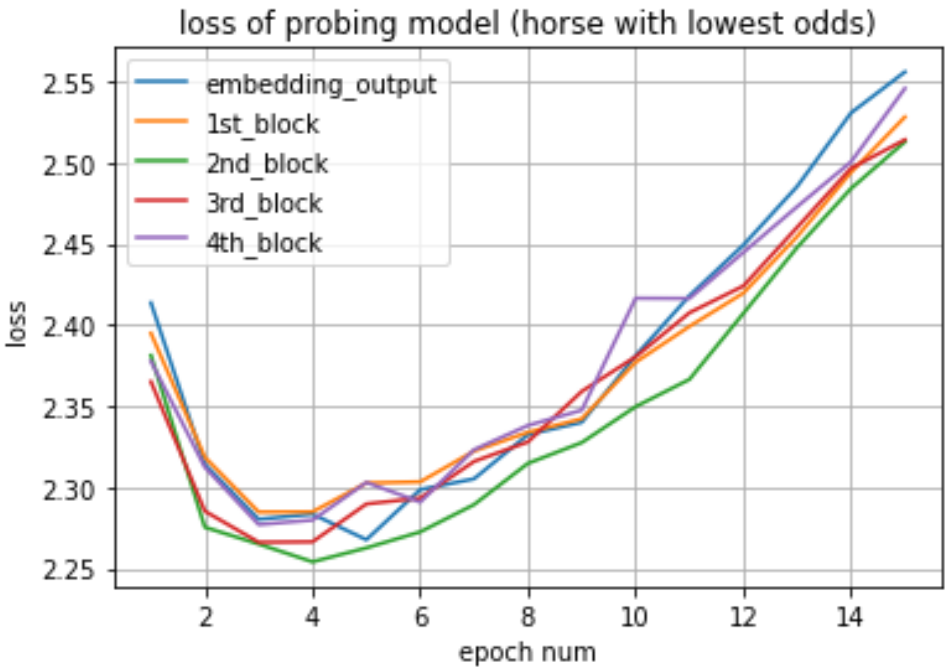


Figure 46. Loss of probing model (horse with lowest odds)

The internal vector representation of the second encoder layer has the lowest loss among all layers, as shown in Figure 46. Furthermore, the loss of the third encoder layer is lower than all other layers except the second layer. Since the concept of the

most popular horse includes a comparison between horses' attributes and it is complex, we expect the later layers to learn better at the abstract idea and they always perform better than the earlier layer. Nonetheless, the result in Figure 46 does not entirely support it because the third and the second layers have higher losses than that of the second layer. This concludes that the later layers usually grasp the abstract concept better, but it is not a must in the horse racing context.

6.1.4 Usefulness of Ratings

Although the rating is an important factor in winning horse prediction as it summarizes a horse's overall performance based on its past records, there are some cases in which the winning horse does not have a high rating. In these cases, the rating is useless, and the prediction should not depend on the rating. We want to know whether the model could determine the usefulness of rating and make use of it in prediction.

The input in the probing dataset is the same as the input in section 6.1.2. The target of the dataset is a truth value about the usefulness of rating, which has True and False as the value. The target y of a race is an integer 0, indicating that the rating is useless if the horses with low ratings win the first three places. Otherwise, y is an integer 1, indicating that the rating is useful.

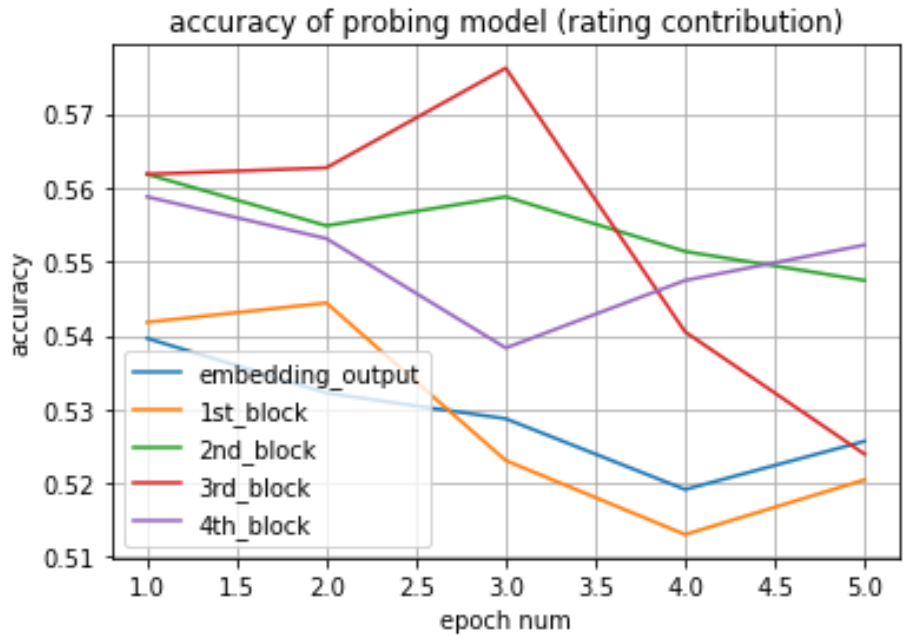


Figure 47. Loss of probing model (rating contribution)

In the classification of the rating usefulness, the internal vector representation at the third layer shows the best performance of attaining 57.7% accuracy. The vector representations at the second and fourth layers both get 55.9% accuracy while the first layer and the embedding out result in 54.5% and 54% respectively, as shown in Figure 47. The highest accuracy obtained is 57.7% which is slightly better than making a correct random guess between useful and useless of 0.5 probability. One reason for the result is the complexity of deciding the usefulness of a rating. Suppose the model fully understands the usefulness of rating in the race. In that case, it can just give a prediction directly relying on the rating, and the expected accuracy of our transformer model would be much higher. However, this task is very complex, and our transformer model has only 23.4% accuracy in the correct prediction of the winner from Figure 40.

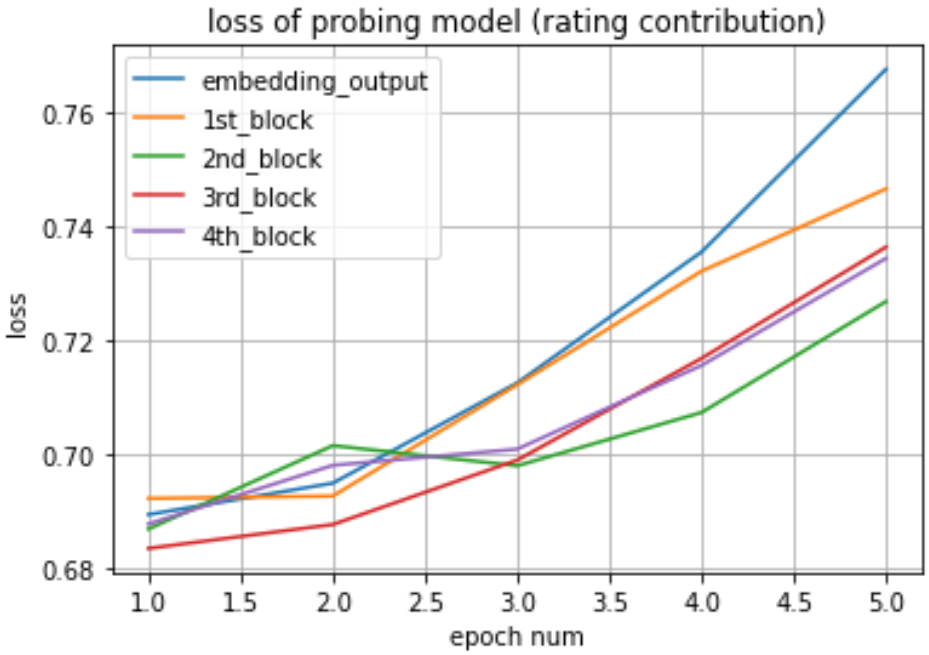


Figure 48. Loss of probing model (rating contribution)

As detecting the usefulness of rating is a very complex task, our assumption states that earlier layers are not as capable as the later layers in this task because earlier layers learn only simple features. In Figure 48, it shows that the claim is valid to a certain extent because the embedding output and the first encoder layer, which are the earlier layers, produce higher loss when their internal vector representations are used for finding the usefulness of rating than that of the second, third and fourth layers.

6.2 Better Performance with Ascending Horse Number

The input of the transformer model is a sequence of horse tokens arranged in ascending order according to the horse number. Therefore, a horse with horse number 1 is positioned on the first horse token, and the horse with horse number 2 is arranged on the second horse token. The other horses are placed in the same way. We want to inspect the performance change of our model in a situation where the horse tokens in the input are reordered randomly so that they are no longer in ascending order. In Figure 49, the model's accuracy drops by 1.7%, from 23.4% to 21.7% after data shuffling. Also, the loss of the model before the shuffling is obviously lower than that after the shuffling, as shown in Figure 50. We will explain this phenomenon from the perspectives of distribution in the dataset and the attention map.

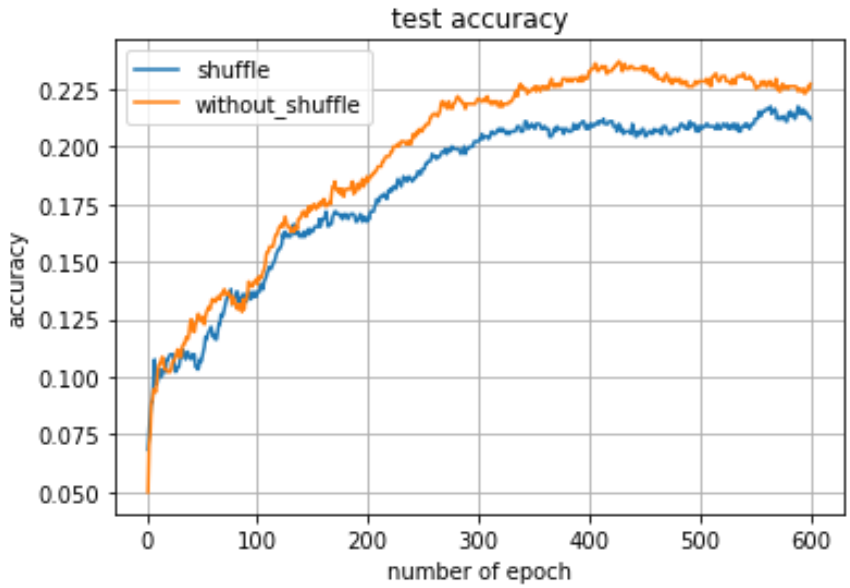


Figure 49. Comparison of accuracy before and after data shuffling

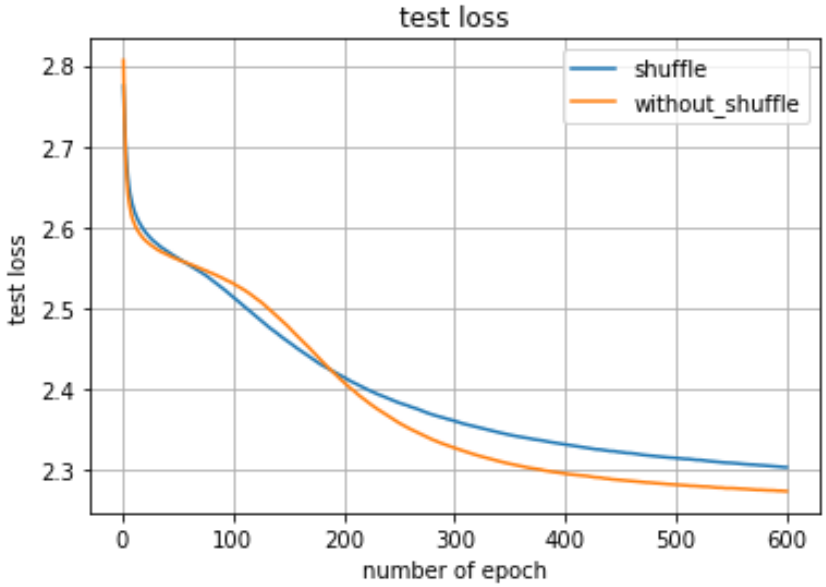


Figure 50. Comparison of loss before and after data shuffling

6.2.1 Distribution of Horses with the Lowest Odds

Arranging the horse tokens in ascending order in terms of horse number implies hidden information about the probability of winning for horses. Figure 51 shows the distribution of horses with the lowest odds. If the horse has the lowest odds in a race, it is the most popular horse perceived by the public, and this information was proven helpful in prediction [9]. The model may learn the distribution in Figure 51 after the training and it may know that the probability of the horse with the lowest odd decreases as the horse number increases. This hidden information may induce bias in the model, and it tends to guess horses with a smaller number to be the winner more often. If the horse tokens are rearranged randomly, the model may not learn the negative relationship between the horse's probability with the lowest odds and the horse number. It loses this information, and the predicting power may be undermined.

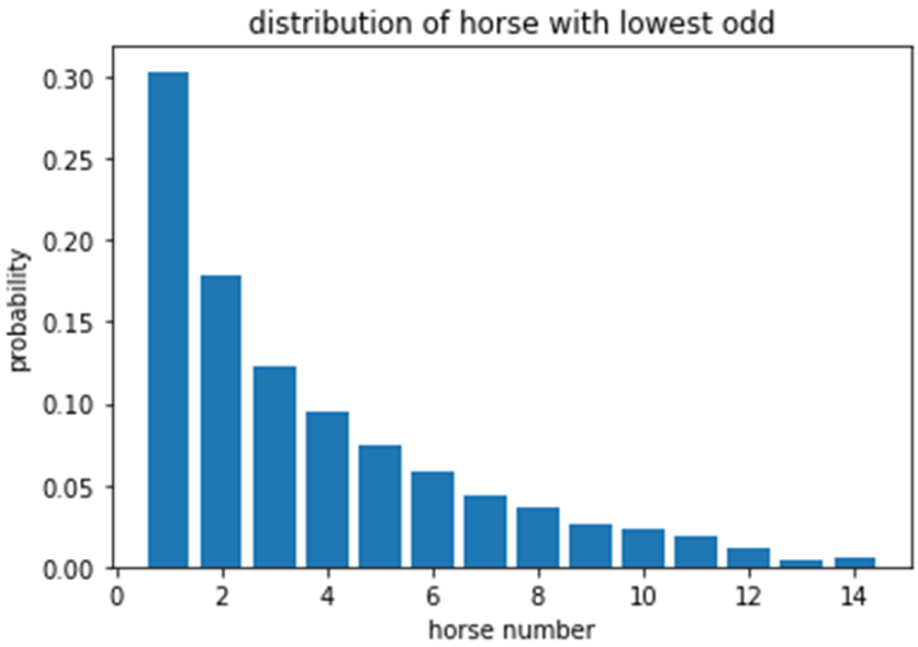


Figure 51. Distribution of horses with lowest odds.

6.2.2 Properties of Attention Map in Successful Transformer Model

Since we attempt to reduce the horse racing classification to a language processing classification by using the transformer and embedding, we expect the attention map of our transformer model will be similar to that of a successful transformer model such as BERT if our model performs well. It suggests a comparison of the attention map in our model with that in BERT to evaluate our model performance. Therefore, we will analyze the attention maps of our model before and after data shuffling based on the properties of the attention map in BERT.

From the observations of the attention map in BERT, we concluded that there are four general properties of a good attention map [39]. The first property is the appearance of recurring patterns in attention heads. The second property is the similar behaviors of heads in the same layer. The third property is the little attention on the same token in most heads. The fourth property is the broad attention of heads in lower layers. The attention maps before and after data shuffling are visualized by an open-source tool [40]. These four properties will be used to evaluate our attention maps and explain the better performance of using horse tokens with ascending order as the input.

6.2.3 Attention Map Evaluation

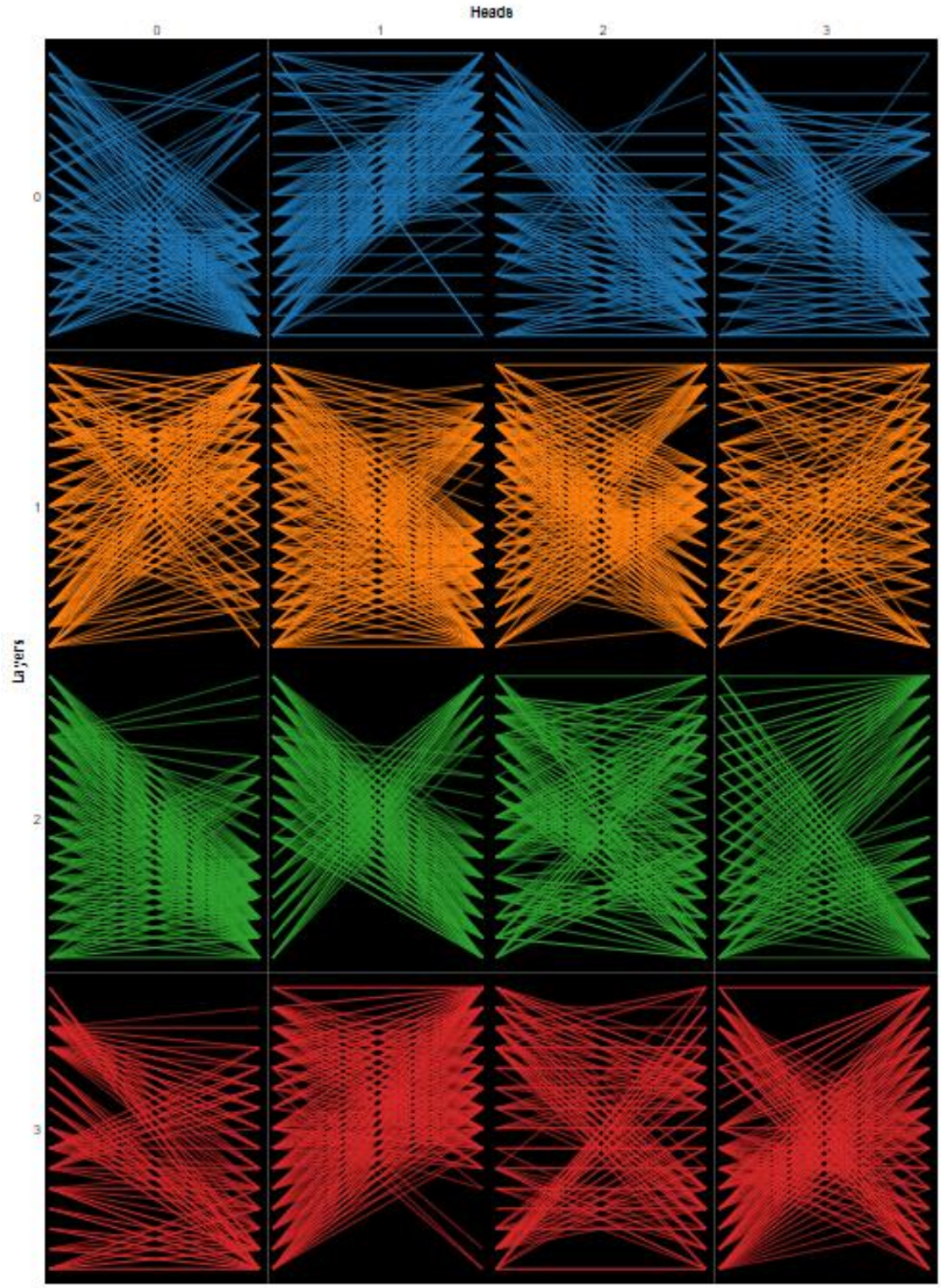


Figure 52. Attention map trained by data before shuffling

The attention map trained by data before in Figure 52 is considered with reference to

the four properties mentioned in section 6.2.2. Firstly, the appearances of recurring patterns in attention heads are recognized. For example, the fourth head in the first layer, the second to fourth heads in both the second and third layers, and the fourth head in the last layer exhibit a recurring pattern in which the first few tokens tend to pay more attention to the last few tokens. In comparison, the last few tokens tend to pay more attention to the first few tokens, and a cross pattern is likely to form. Secondly, we observe similar behaviors for heads in the same layer. All heads in the first layer, second and fourth, tend to pay strong attention to tokens further away from the current tokens. All heads in the second layer tend to pay attention broadly to every token, satisfying the fourth property simultaneously. Also, the third property is satisfied because there are only a few numbers of heads having weak attention to the same token. Therefore, the attention map trained by data before shuffling has all properties of the attention map in a successful transformer model.

The existence of those four properties is examined in the attention map trained by data after shuffling. Firstly, a recurring pattern of paying strong attention to tokens in the first few and last few tokens exists in some heads such as the first and second heads in the first layer, the first head in the second layer, and the third and fourth heads in the third layer. However, the second property does not exist because all heads in the first layer tend to behave differently. Also, the first head in the third layer exhibits broad attention while the second head does not. The third property is not satisfied because many heads pay attention to the same tokens. For example, the third and fourth heads in the first layer, the fourth head in the second layer, the second head in the third layer, and the first, second, and third heads in the fourth layer. The fourth property is also unsatisfied because broad attention is only observed in the third layer's first head, which disagrees with the fourth property that there is broad attention of heads in the

lower layer.

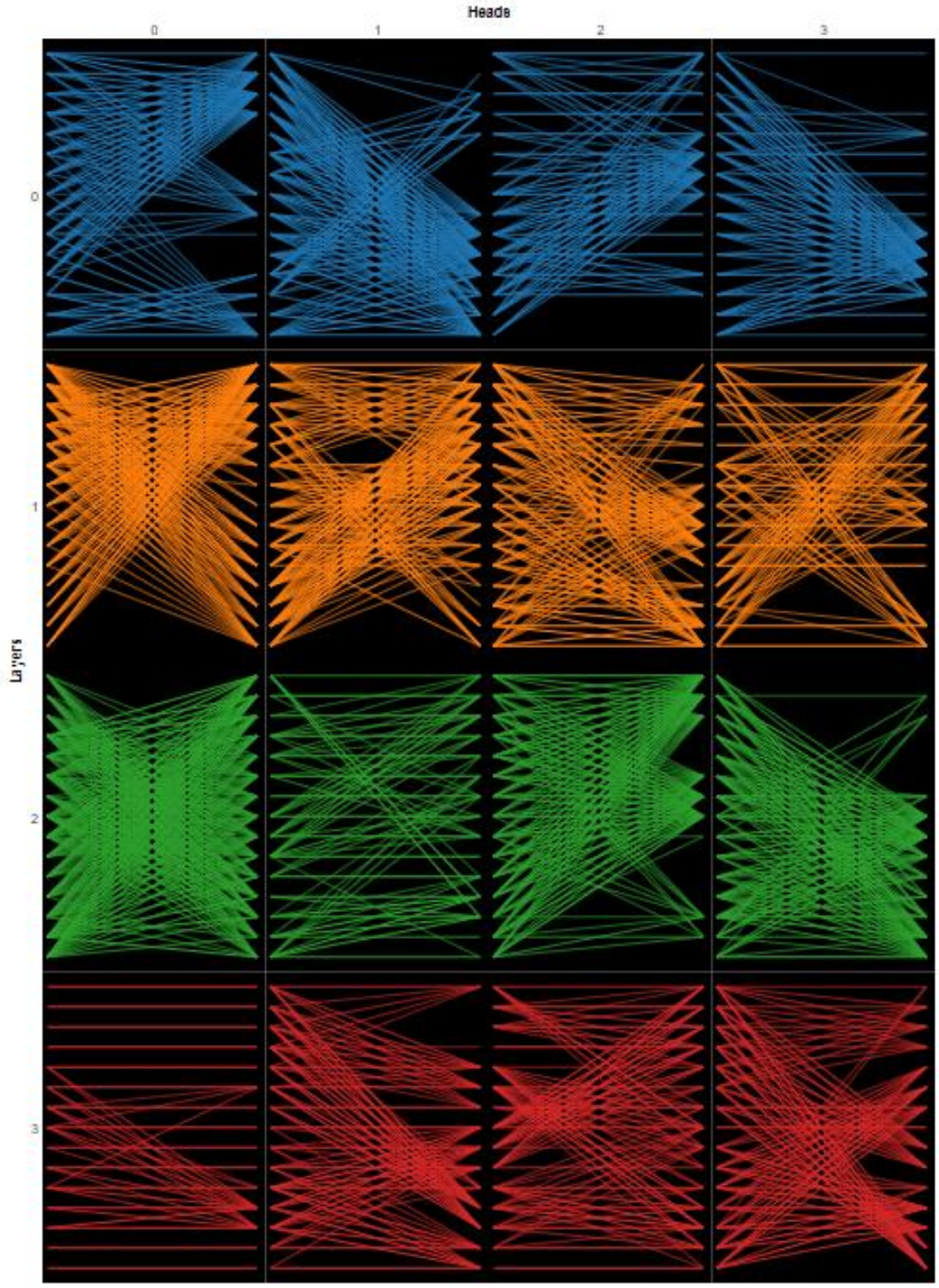


Figure 53. Attention map trained by data after shuffling.

6.2.4 Explaining the Results

Our investigations of the phenomenon of the better performance of the model when the horse tokens input is in ascending order provide two possible reasons. The first reason is that the model can learn the negative relationship between the probability of being the most popular horse and the horse number if the horse tokens are in ascending order. This information may lose after the shuffling of the horse tokens.

The second reason is that the shuffling of horse tokens leads to the poorer behaviors of the attention map as the attention map trained before the data shuffling has all properties of the attention map in a successful model while that after the data shuffling possesses only one of the properties. The comparison can be found in Table 6.

	Attention map trained by data before shuffling	**Attention map trained by data after shuffling**
Recurring pattern	✔	✔
Similar behavior in the same layer	✔	✘
Little attention to the same token	✔	✘
Broad attention in lower layers	✔	✘

Table 6. Existence of properties of successful attention map

6.3 Contribution of Horse Tokens to the Prediction

Our transformer demonstrates its capability in making a profit in betting simulation, and this means that the model generalizes some ideas in the learning process, which provides profitable prediction. We want to look at those ideas and conceptualize them into simple rules that assist the betting. The integrated gradient is chosen to be the tool for us to realize the input-output behavior of the model and determine the importance of each horse token [41].

Before the investigation, we have prior assumptions below about the contributions of horse tokens to the prediction, which will be verified in the analysis. Suppose the horse with horse number i is the predicted winner, we expect it should have the highest positive contribution to the prediction. Horses other than the winner should give negative contributions because they are competitors of the winner. Besides, the contributions of the strong horses should be more significant because they are more influential in the race.

6.3.1 Integrated Gradient

We select the integrated gradients method because we can utilize the gradient operations to compute the integrated gradients by integrating the first-order derivatives with little effort while maintaining our model architecture [41]. After the computation, the input features' attribution of the model prediction can be obtained for further analysis. Hence, integrated gradients can approximate the importance of horse tokens in the race.

Suppose the input sequence of horse tokens for a race is x and the baseline input is x'. We denote our transformer model as F, and the model's output is $F(x)$. The gradients of all points along the straight-line path from the baseline x' to the input x are integrated to obtain the integrated gradient [42]. Therefore, we can use the following equation to see how the m^{th} horse token in the input sequence x contributes to the model prediction $F(x)$,

$$IntegratedGradient_m(x) = (x_m - x_m') \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_m} d\alpha. \quad (18)$$

6.3.2 Contributions in Different Situations

In this section, the contributions of all horse tokens in different situations are computed for extraction of commonalities, and every situation has a distinctive winner. The positive contribution of the horse token is colored in blue, while the negative contribution is colored in red.

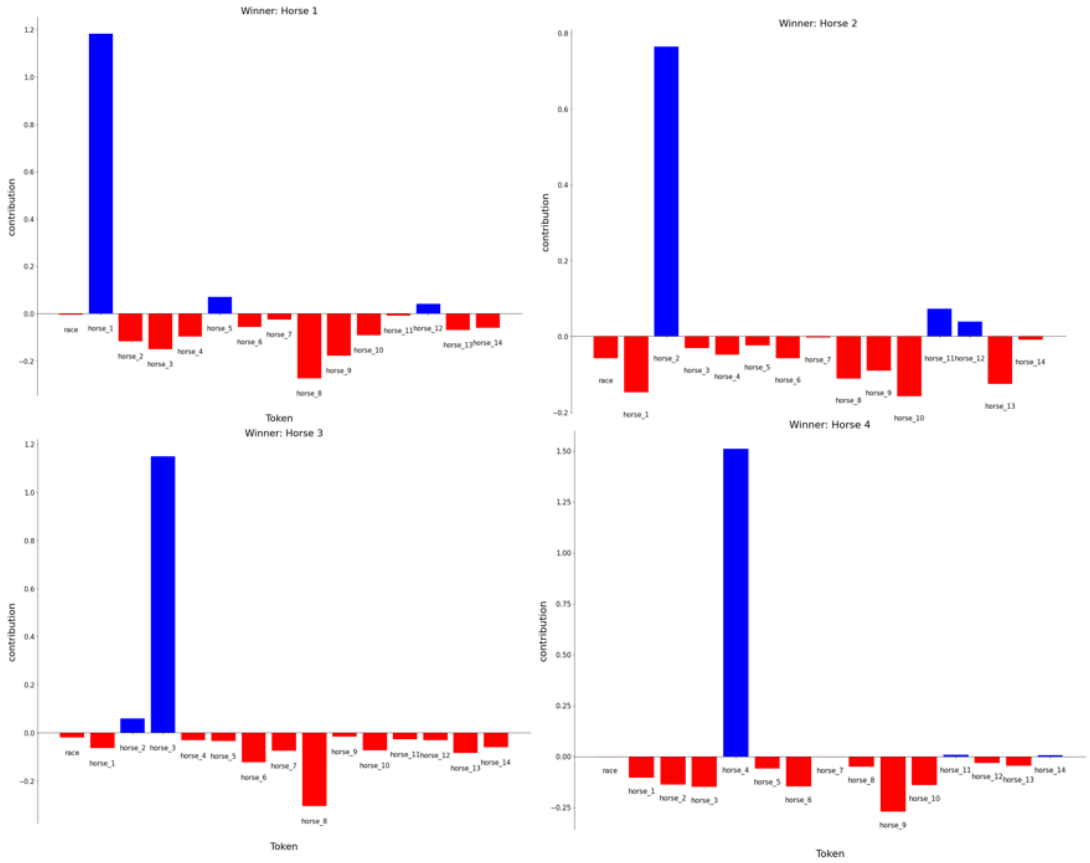


Figure 54. Contributions of horse tokens when horses 1 – 4 are winners

When the winners are a horse with numbers 1 to 4, the winner contributes positively to a large extent and most other horses contribute negatively as seen in Figure 54. We notice that horses with numbers 8, 9, and 10 contribute a significant negative value while the remaining horses contribute relatively low negative values.

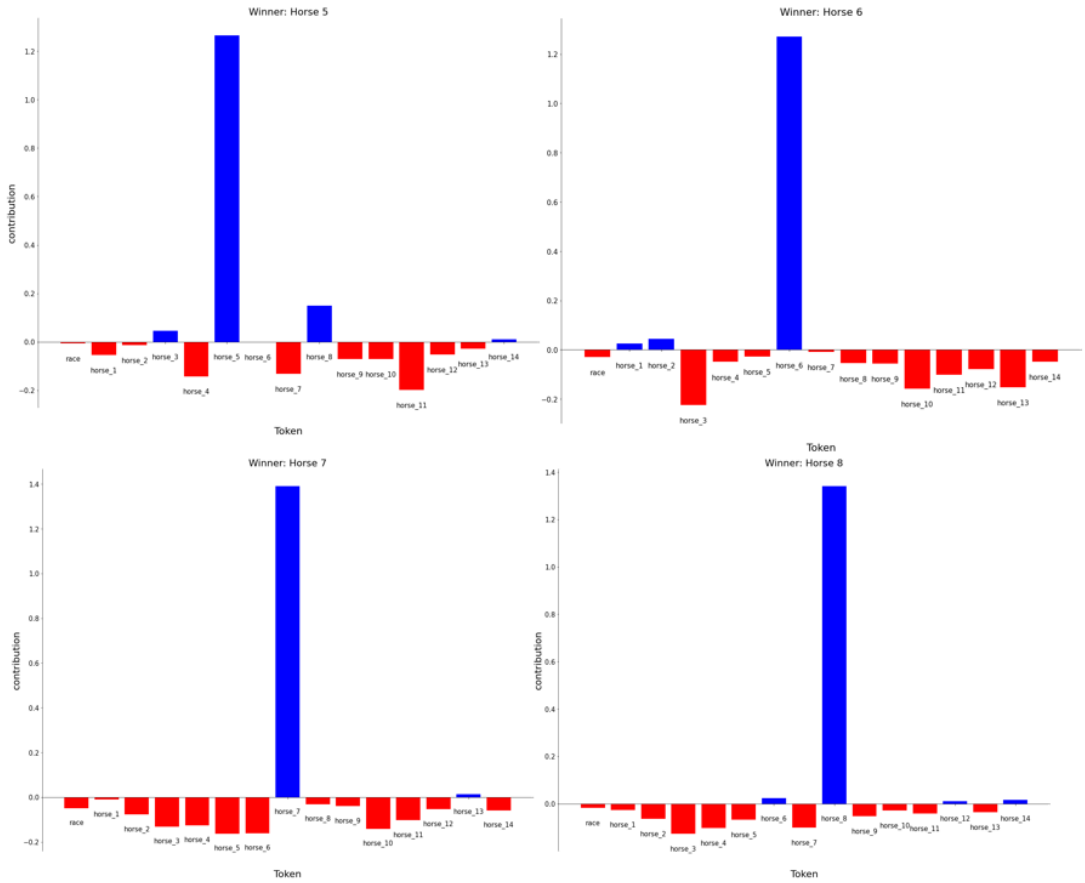


Figure 55. Contributions of horse tokens when horses 5 – 8 are winners

For races whose winners are horses with a number from 5 to 8, the winner contributes a significant amount of positive value, and most other horses contribute negatively. In Figure 55, we discover that the horse that contributes most negatively are not the horses next to the winner but the horses further away from it. For example, if the winner is horse number 5, horse number 11 has the most negative value instead of horse number 4 or number 6.

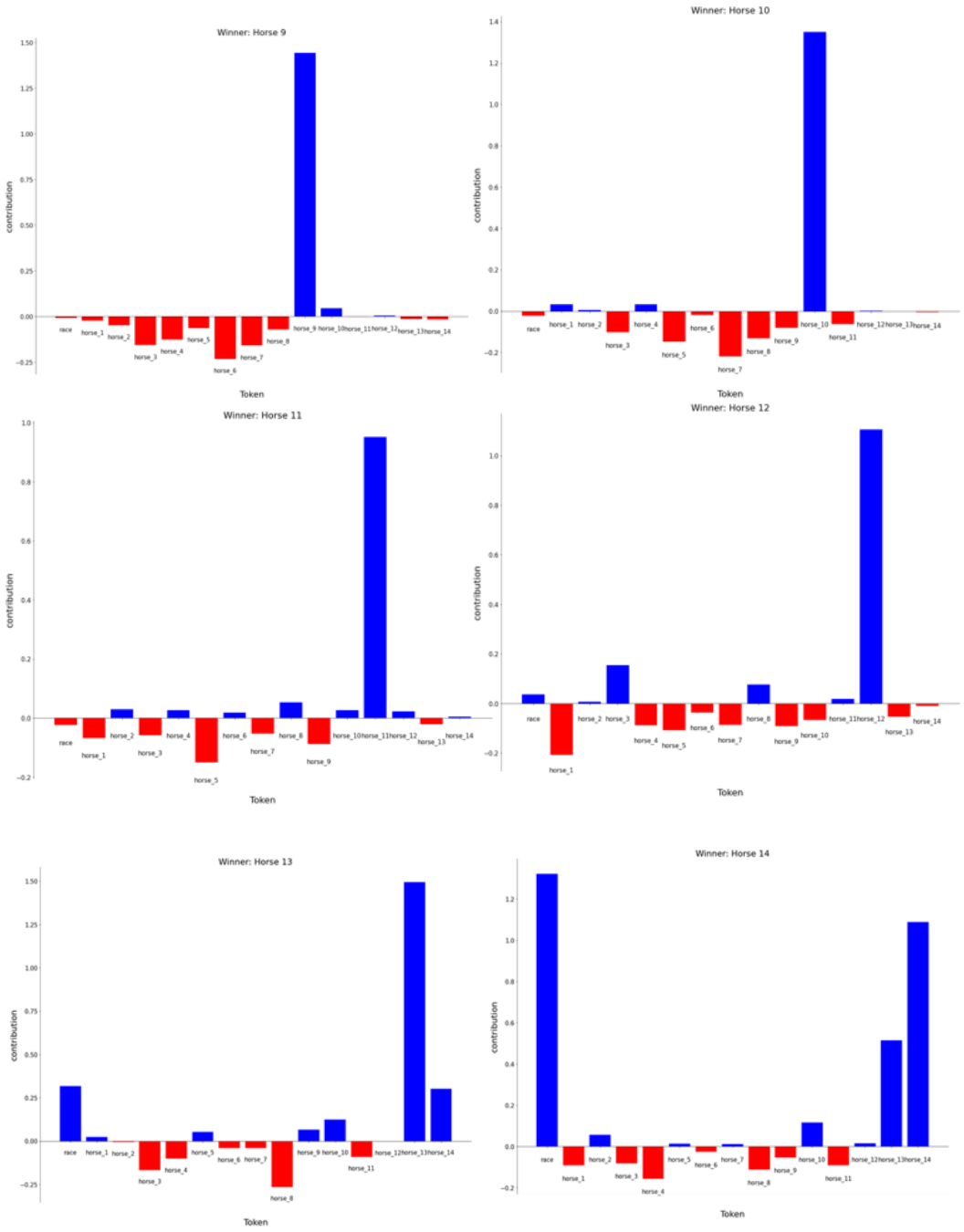


Figure 56. Contributions of horse tokens when horses 9 – 14 are winners

For races whose winners are horses with a number from 9 to 14 in Figure 56, we observe that the contribution patterns for winners from 12 to 14 are different. For these three cases, the input regarding the race information contributes a positive value while it contributes a negative value for all the other cases. Besides, the number of horses giving positive contributions increases when the winners are a horse with a

number from 12 to 14.

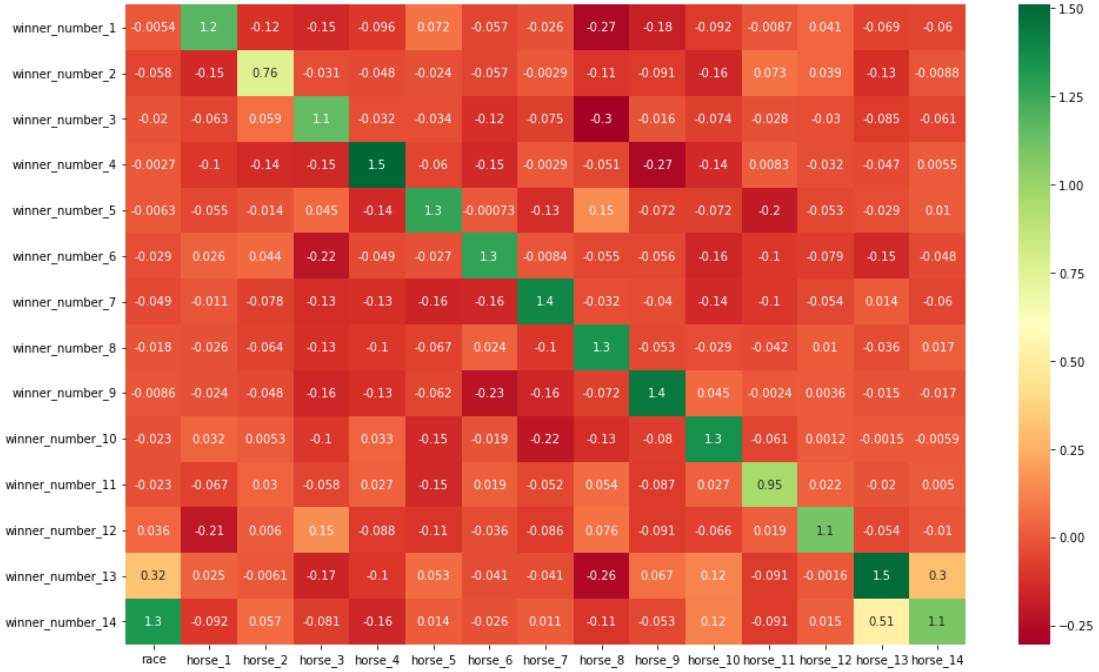


Figure 57. Heatmap showing contributions of horse tokens in all situations

6.3.3 Simple Rules Extraction

The advantage of the integrated gradient in model inspection is showing the mapping of input-output behavior [41]. We can often gain insights from the visualizations of the mapping to generalize a set of simple rules that help us predict the output. From the results obtained in section 6.3.2, we conclude the following simple rules that guide the betting people.

1. Given that you want to bet on a horse with horse number i for $1 \leq i \leq 14$, you should focus on the horses far from horse i and consider their ratings. If those horses' ratings are high, horse i is not likely to win. This rule is based on the observation that the negative contribution of horse j with $|i-j|$ which indicates that the negative impact of horse j will likely be enlarged when it is distant from horse i

2. Given that you want to bet on horse i for $12 \leq i \leq 14$, you should consider the race condition. If horse i performed well in a similar race condition in the past, the probability for horse i to be the winner is large because we observe that the race condition contributes a significant amount of positive value when the winners are horses with a number greater than 11.

3. Given that you want to bet on horse i for $12 \leq i \leq 14$, a few strong horses with a horse number smaller than i should not discourage you from betting on horse i. We discover that some horses with numbers smaller than i contribute positively when the winner is i. One possible explanation is that those horses are strong that they block the other horses in the race and leave chances for horse i to catch up with them.

Chapter 7

Conclusion

The first phase of the project aims at understanding the betting odds in horse racing by analyzing the impact of using both the rating systems and transformer architecture on the accuracy and profit-making aspects of horse racing prediction. From previous studies, the win odds in the feature list enhance the accuracy and net gain [8][9]. We exclude the win odds from the feature list this time and attempt to resemble the effect of the win odds by combining the performance judgment and natural language processing techniques.

We contrast the differences in performance by experimenting with three models: multilayer perceptron with ratings, transformer without ratings, and transformer with rating. We discover that the best case of our models is the transformer with Elo-MMR ratings, which has the highest test accuracy of 21.4% and gives a positive net gain of 6% in the betting simulation of the test data. This shows that ratings and transformer architecture have similar influences on the horse racing prediction.

The second phase of the project extends the work of the first phase from two perspectives. The first perspective focuses on accuracy improvement. When applying the transformer architecture in the horse racing context, a more appropriate horse token embedding is suggested to replace the word embedding. The horse token embedding can be efficiently generated with proper data segregation and the aid of Principal component analysis. The accuracy of the best model chosen in the first phase is boosted by 2%, from 21.4% to 23.4% after replacing the embedding.

The other perspective shifts the attention to the interpretability of the best model found in the first phase. Three different aspects are investigated using different interpretability methods. The first one is the assessment of the model's capabilities in learning helpful information by probing. It exposes that later layers are usually more capable of discovering useful information. The second aspect is the impact of input order perturbation on the model performance. It shows that the perturbation causes the loss of information and poor attention heads' behaviors, which results in worse model performance. The third aspect is studying the input-output behaviors of the model by the integrated gradient. Simple rules are extracted from the input-output behaviors to guide the betting.

References

[1] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, Perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[2] E. Alpaydin, “What Is Machine Learning?,” in *Introduction to machine learning*, Cambridge, MA: The MIT Press, 2014.

[3] P. Cunningham, M. Cord, and S. J. Delany, “Supervised learning,” *Machine Learning Techniques for Multimedia*, pp. 21–49.

[4] S. Becker, “Unsupervised learning procedures for neural networks,” *International Journal of Neural Systems*, vol. 02, no. 01n02, pp. 17–33, 1991.

[5] El Naqa and M. J. Murphy, “What is machine learning?,” *Machine Learning in Radiation Oncology*, pp. 3–11, 2015.

[6] Prieto, M. Atencia, and F. Sandoval, “Advances in artificial neural networks and machine learning,” *Neurocomputing*, vol. 121, pp. 1–4, 2013.

[7] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, “Transformers: State-of-the-art natural language processing,” *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020.

[8] T. T. Cheng and M. H. Lau, "Predicting Horse Racing Result using TensorFlow," Department of Computer Science and Engineering, Hong Kong, 2017.

[9] Y. Liu and Z. Wang, "Predicting Horse Racing Result with Machine Learning," Department of Computer Science and Engineering, Hong Kong, 2018.

[10] Y. Wong, "Horse Racing Prediction using Deep Probabilistic Programming with Python and PyTorch (Uber Pyro)," Department of Computer Science and Engineering, 2018.

[11] “RACING IN HONG KONG,” *Hong Kong Racing 101*, Sep-2016. [Online]. Available: https://entertainment.hkjc.com/entertainment/common/chinese/images/more-about-racing/racing-101/Racing-101_201509.pdf. [Accessed: 13-Nov-2021].

[12] “The Hong Kong jockey club,” *About HKJC - The Hong Kong jockey club*. [Online]. Available: https://corporate.hkjc.com/corporate/english/index.aspx. [Accessed: 13-Nov-2021].

[13] R. H. Thaler and W. T. Ziemba, “Anomalies: Parimutuel betting markets: Racetracks and lotteries,” *Journal of Economic Perspectives*, vol. 2, no. 2, pp. 161–174, 1988.

[14] N. Silverman and M. Suchard, “Predicting horse race winners through a regularized conditional logistic regression with frailty,” *The Journal of Prediction Markets*, vol. 7, no. 1, pp. 43–52, 2013.

[15] “Investments and Dividends,” *The Hong Kong Jockey Club*. [Online]. Available: https://special.hkjc.com/infomenu/en/info/investments.asp. [Accessed: 14-Nov-2021].

[16] “Betting Guide,” *Pari-mutuel local pools - beginners guide - betting entertainment - the hong kong jockey club*. [Online]. Available: https://special.hkjc.com/racing/info/en/betting/guide_qualifications_pari.asp. [Accessed: 13-Nov-2021].

[17] C. Yau, “Hong Kong jockey club posts record HK$280 billion revenue despite pandemic,” *South China Morning Post*, 31-Aug-2021. [Online]. Available: https://www.scmp.com/news/hong-kong/hong-kong-economy/article/3147062/hong-kong-jockey-club-posts-record-revenues-hk280. [Accessed: 13-Nov-2021].

[18] Davoodi, Elnaz, and Ali Reza Khanteymoori. "Horse racing prediction using artificial neural networks." *Recent Advances in Neural Networks, Fuzzy Systems & Evolutionary Computing* 2010 (2010): 155-160.

[19] R. N. Bolton and R. G. Chapman, "Searching for positive returns at the track: A multinomial logit model for handicapping horse races," *Management Science*, vol. 32, no. 8, pp. 1040–1060, 1986.

[20] S. Lessmann, M.-C. Sung, and J. E. V. Johnson, “Alternative methods of predicting competitive events: An application in Horserace Betting Markets,” *International Journal of Forecasting*, vol. 26, no. 3, pp. 518–536, 2010.

[21] W.-C. Chung, C.-Y. Chang, and C.-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.

[22] GLICKO (Glickman, Mark E. "The glicko system." *Boston University* 16 (1995): 16-17)

[23] Herbrich, Ralf, Tom Minka, and Thore Graepel. "Trueskill™: A Bayesian skill rating system." *Proceedings of the 19th international conference on neural information processing systems*. 2006.

[24] Ebtekar and P. Liu, “Elo-MMR: A rating system for massive multiplayer competitions,” *Proceedings of the Web Conference 2021*, 2021.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., "Attention Is All You Need", *CoRR*, vol. abs/1706.03762, 2017.

[26] A. Lindholm, “Health aspects of horse production including training with special reference to Nordic conditions,” *Livestock Production Science*, vol. 40, no. 1, pp. 73–76, 1994.

[27] L. Beretta and A. Santaniello, “Nearest neighbor imputation algorithms: A critical evaluation,” *BMC Medical Informatics and Decision Making*, vol. 16, no. S3, 2016.

[28] K. Potdar, T. S., and C. D., “A comparative study of categorical variable encoding techniques for neural network classifiers,” *International Journal of Computer Applications*, vol. 175, no. 4, pp. 7–9, 2017.

[29] M. K. Dahouda and I. Joe, “A deep-learned embedding technique for categorical features encoding,” *IEEE Access*, vol. 9, pp. 114381–114391, 2021.

[30] J. Sola and J. Sevilla, “Importance of input data normalization for the application of neural networks to complex industrial problems,” *IEEE Transactions on Nuclear Science*, vol. 44, no. 3, pp. 1464–1468, 1997.

[31] EbTech and P. Liu, “EbTech/Elo-MMR: Skill Estimation Systems for multiplayer competitions,” *GitHub*. [Online]. Available: https://github.com/EbTech/Elo-MMR. [Accessed: 24-Nov-2021].

[32] K. Kira and L. A. Rendell, “A practical approach to feature selection,” *Machine Learning Proceedings 1992*, pp. 249–256, 1992.

[33] B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich, and F. A. Hamprecht, “A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of Spectral Data,” *BMC Bioinformatics*, vol. 10, no. 1, 2009.

[34] “Embedding,” Embedding - PyTorch 1.11.0 documentation. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html. [Accessed: 28-Mar-2022].

[35] F. Murtagh, “Multilayer perceptrons for classification and regression,” *Neurocomputing*, vol. 2, no. 5-6, pp. 183–197, 1991.

[36] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:*1207.0580, 2012.

[37] O. Pandit and Y. Hou, “Probing for bridging inference in Transformer Language Models,” *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[38] D. Johnson, D. Mak, A. Barker, and L. Loessberg-Zahl, “Probing for multilingual numerical understanding in transformer-based language models,” *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2020.

[39] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, “What does Bert Look at? an analysis of Bert’s attention,” *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019.

[40] J. Vig, "Visualizing attention in transformer-based language models", *arXiv preprint arXiv:1904.02679*, 2019.

[41] S. He, Z. Tu, X. Wang, L. Wang, M. Lyu, and S. Shi, “Towards understanding neural machine translation with word importance,” *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[42] Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks." *International conference on machine learning*. PMLR, 2017.