Betting Odds Calculation with Machine Learning

LYU 2102

NAM Man Leung

supervised by Prof. Michael Lyu

Outline

- Brief Summary of 1st Term
- Objectives of 2nd Term
- Model Improvement
- Interpretability
- Conclusion

Brief Summary of 1st Term

Brief Summary of 1st Term

- Win odds contain helpful information in horse racing prediction[1][2]
- Win odds keep changing before the start of the race
- data collection of the win odds may not be precise enough before the start of the race
- betting is not permitted after the beginning of the race
- attempt to use only static data which do not vary within the betting period

Target: Achieve satisfactory performance with the exclusion of win odds

Replacement of Win Odds by Rating

1. The Glicko Rating System[3]
2. The TrueSkill Rating System[4]
3. The Elo-MMR Rating System[5]

- The rating encapsulates a horse's ability by a single number based on its past performance
- The rating does not change during the betting period

Model and Rating Comparison

	Multilayer perceptron	Transformer
The Elo-MMR Rating	18.2%	21.4%
The TrueSkill Rating	19.6%	19.4%
The Glicko Rating	20.4%	20.1%

Table 1. Model and Rating Comparison (Accuracy)

Transformer + The Elo-MMR rating

- best combination

Objectives of 2nd Term

Objectives of 2nd Term

1. Model Improvement
    - Change of Embedding method
2. Investigation about Interpretability of the model
    - Assessment to Model Capability
    - Performance Change due to data Shuffling
    - Horse Token Contribution to the Prediction

Model Improvement

Embedding Method Used in the First Term

- word embedding layer from PyTorch[6]
  - all words have to be first encoded into an integer value
  - The encoded words are then fed to the word embedding layer
  - Every word is described by real values after embedding

```
>>> 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 1. word embedding layer from Pytorch[6]

Change of Embedding Method

- Inappropriateness of Word Embedding Layer
  - values 1098, 1103, 1331 may be encoded into integers of values 1, 2, 3
  - 1331 is much greater than 1103 and 1103 is slightly larger than 1098
  - this relationship may not be well captured after encoding
  - 1331 – 1103 > 1103 – 1098
  - 3 – 2 = 2 - 1

horse_1_jockey_name	horse_1_trainer_name	horse_1_actual_weight	horse_1_declared_weight	horse_1_finish_time
DWhyte	JSize	133	1098	01:09.9
CWWong	CHYip	128	1103	01:36.3
DWhyte	ALee	129	1331	01:10.0

Figure 2. Example of Input Data

Change of Embedding Method

- Can we skip the encoding?
- Word Embedding Simulation by Horse Token
  - A horse is similar to a word
  - Every horse is described by real values
  - All features regarding to horse i are used to produce that real values

```
>>> 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 3. word embedding layer output from Pytorch[6]

```
[
  [ 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 4. Example of Horse Tokens

Horse Tokens Generation by PCA

- Partition the horse racing dataset into 14 matrices
- The matrix i contains only attributes of the horse i in all 9191 races
- The size of matrix i was 9191 x n where n is the number of attributes of horse i
- PCA is utilized to reduce the dimensionality of every matrix from n to m.
- All 14 matrices are concatenated horizontally after PCA
- a matrix of size 9191 x 14 x m is obtained

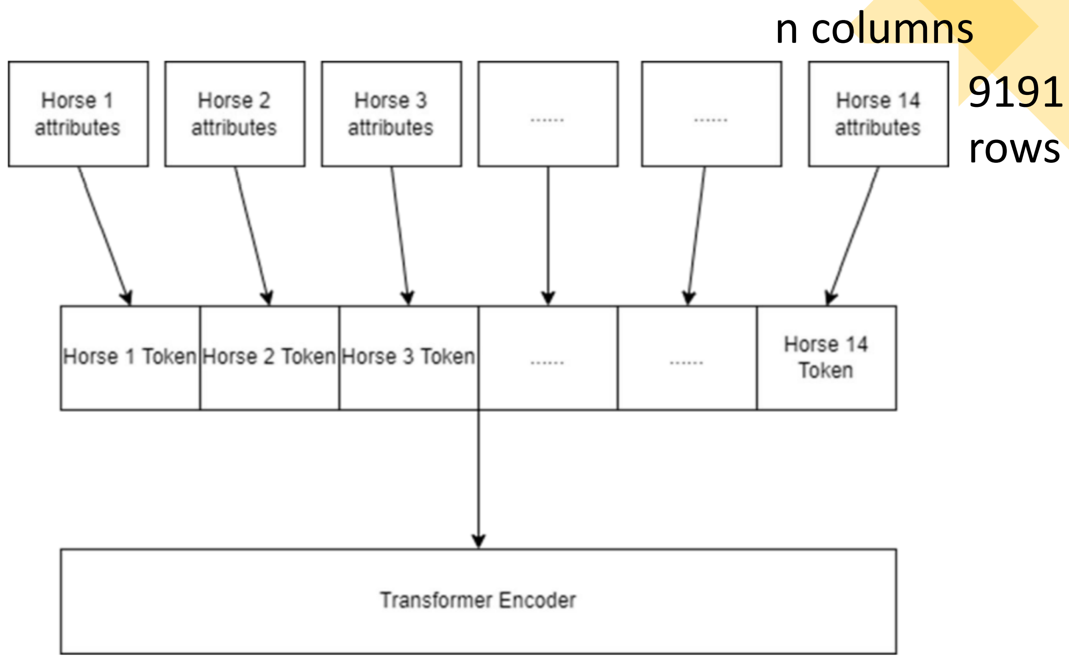

Figure 5. Horse Tokens Generation by PCA

Change of Embedding Method

- Accuracy increases from 21.4% to 23.4%

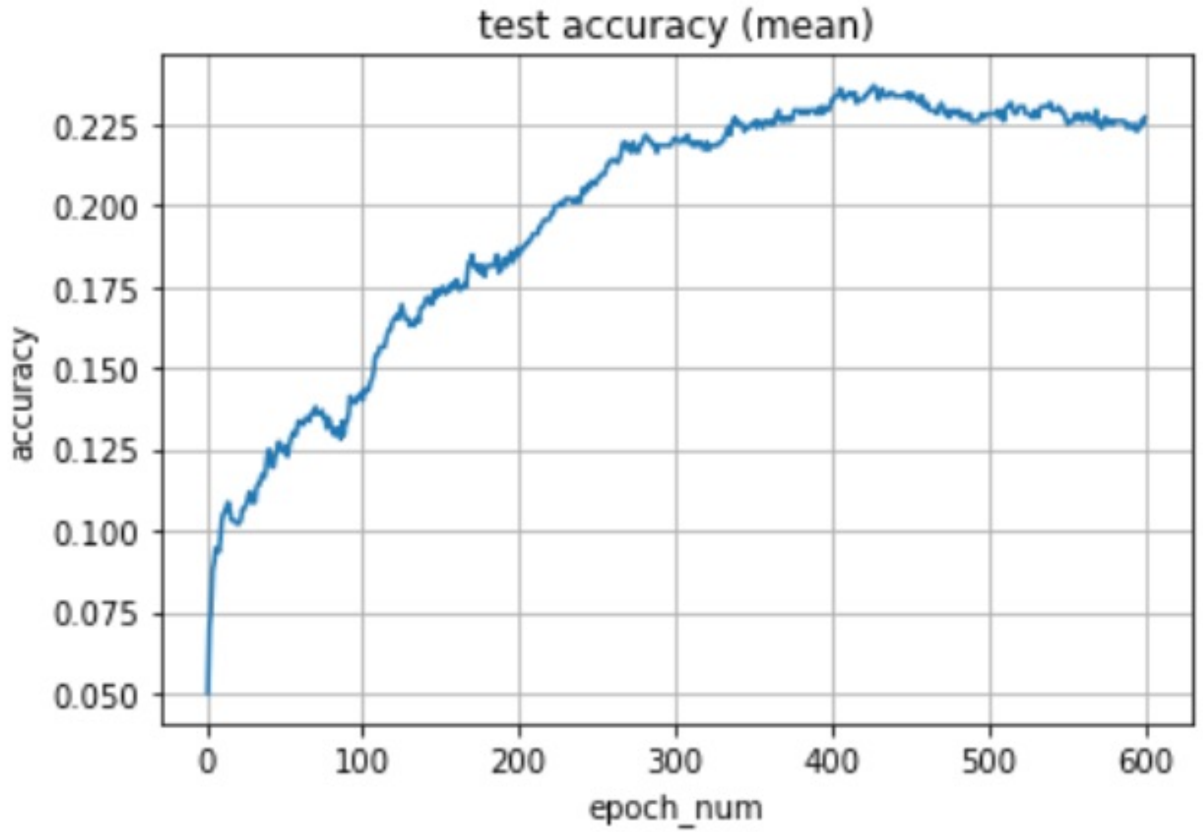

Figure 6. Test accuracy when horse token embedding was used

Change of Embedding Method

- the net gain over the races in test cases has a gentle trend of increase
- the profits from the correct predictions compensate for the losses caused by the wrong predictions

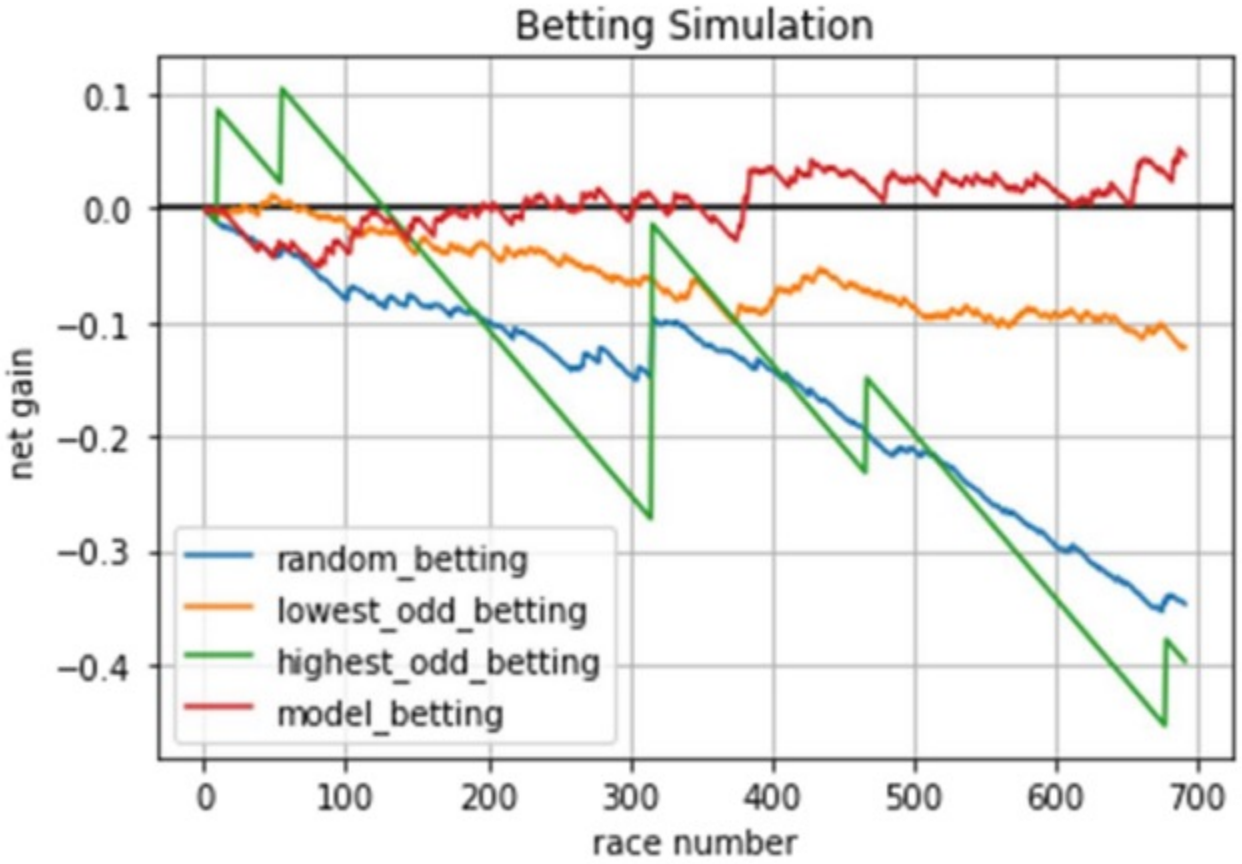

Figure 7. The betting simulation of transformer with horse token embedding

Interpretability
Assessment to Model Capability

Assessment to Model Capability

- use probing tasks for reasoning the capabilities of the transformer understanding the race[7]
- examine the information carried by the internal vector representation of every layer of the transformer encoder
- Feed the internal vector representation to a probing model to see the model's capability

Probing Model

- transformer model has the ability to learn the property of the race if its internal vector representation can help produce accurate prediction about the property

- accurate predictions by the probing model means the information in the internal vector representation is sufficient

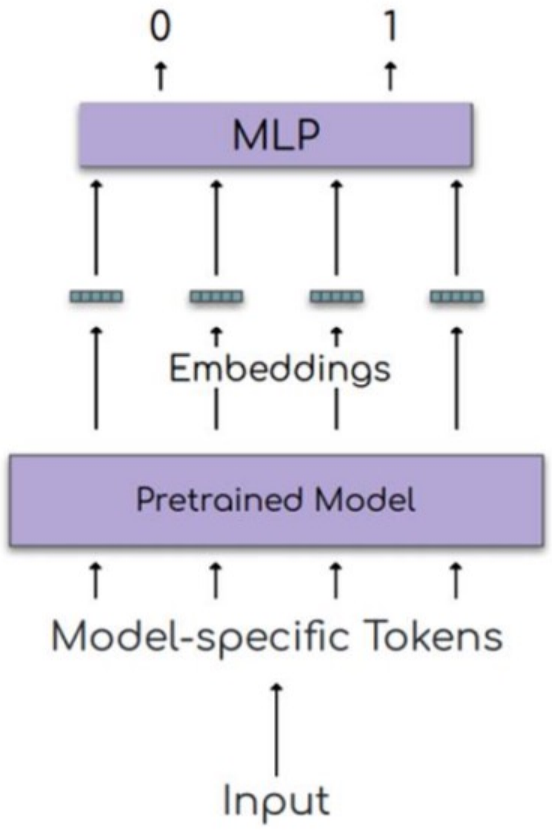

Figure 8. The probing model[7]

Number of Participants

- The number of participants varies in races, and it is usually in the range of 10 to 14.
- the model is expected to know the number of the participant when predicting the winning horse so that the prediction of the winning horse number is within the range

- probing dataset generation
  - the count of participants in each race is extracted from the original dataset, and it is marked as the target

Number of Participants

- The accuracy of classifying the number of participants in all layers is at least 86%
- The high accuracy indicates the capability of the transformer to identify the number of participants correctly in most cases

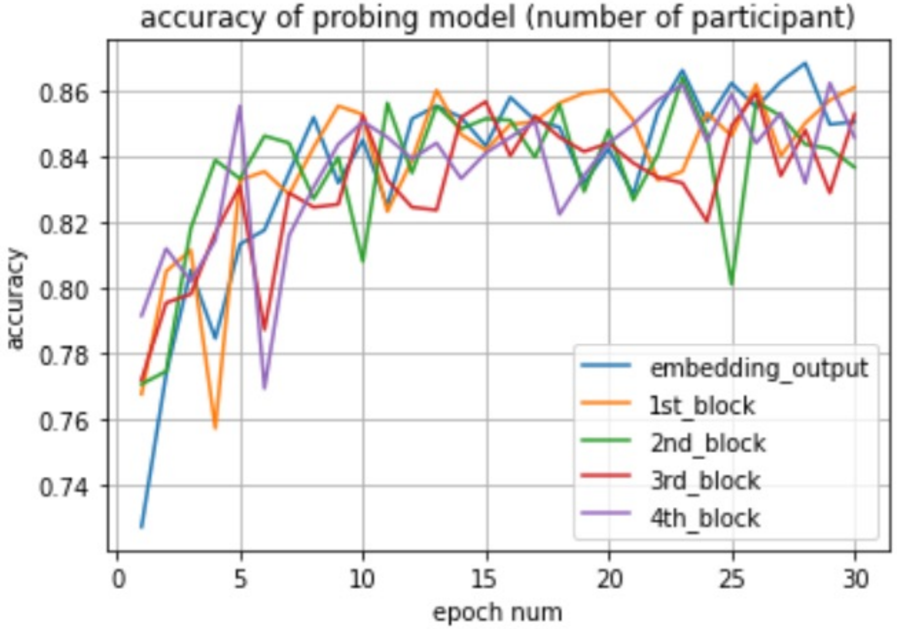

Figure 9. Accuracy of probing model (number of participants)

Most Popular Horse

- win odds could improve the model's performance because it reflects the public intelligence[8]
- most popular horse has the lowest win odds
- win odds are excluded from the input due to its dynamic nature
- determine whether rating and transformer could have a similar impact on finding the most popular horse

- probing dataset generation
    - The target of the dataset is the horse with the lowest win odds

Most Popular Horse

- There are 14 horses in a race, and the probability of selecting the most popular horse in a random guess is 7.14%
- the accuracy is boosted to 26.6% when the internal vector representation was used to assist the selection.
- The model has ability to find the most popular horse in some cases.

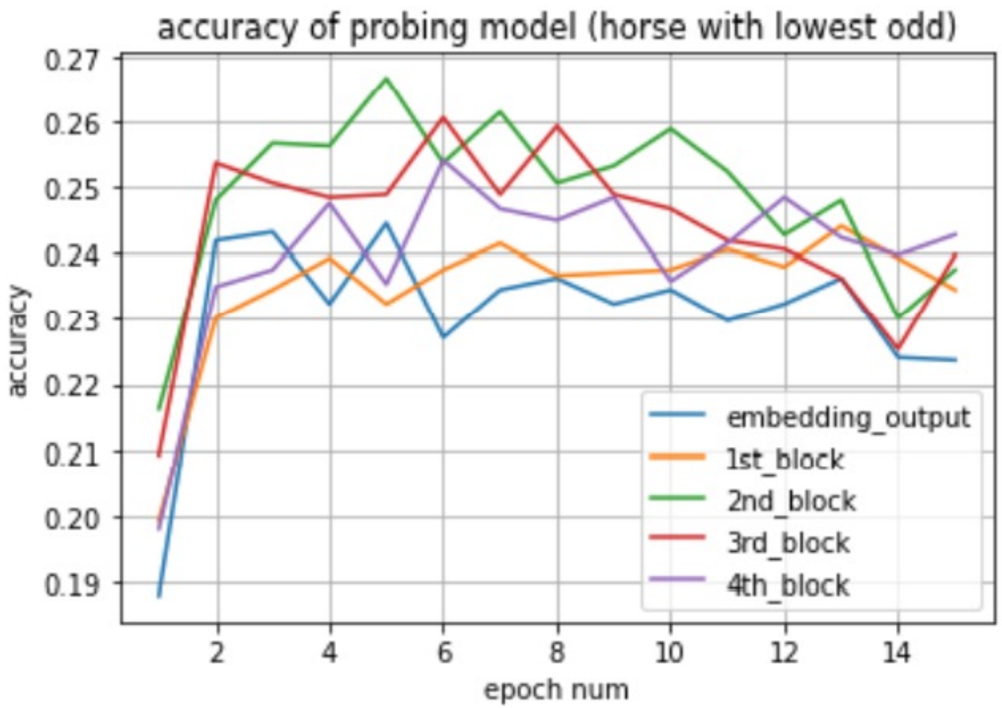

Figure 10. Accuracy of probing model (most popular horse)

Interpretability
Performance Change due to data Shuffling

Worse performance after data shuffling

- The input of the transformer model is a sequence of horse tokens arranged in ascending order according to the horse number
- horse tokens in the input are reordered randomly so that they are no longer in ascending order
- the model's accuracy drops by 1.7%, from 23.4% to 21.7% after data shuffling

Figure 12. Comparison of accuracy before and after data shuffling

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
- 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

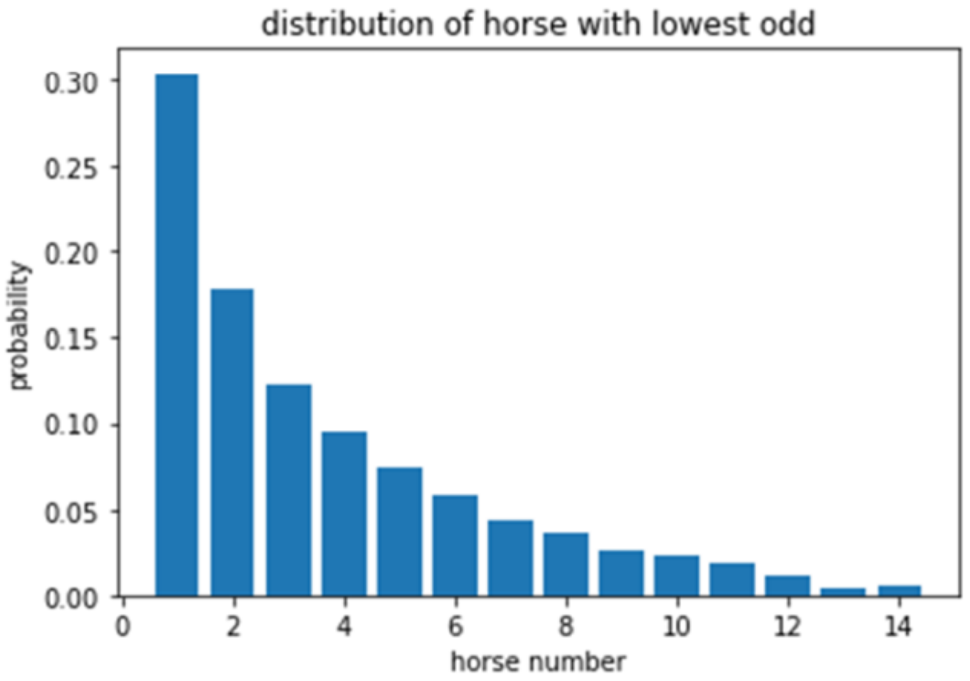

Figure 13. Distribution of horses with lowest odds.

Properties of Attention Map in Successful Transformer Model

- 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
- a comparison of the attention map in our model with that in BERT

- Four general properties of a good attention map[8]
    1. appearance of recurring patterns in attention heads
    2. similar behaviors of heads in the same layer
    3. little attention on the same token in most heads
    4. broad attention of heads in lower layers

Attention Map Evaluation

	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 2. Existence of properties in attention map (before shuffling)

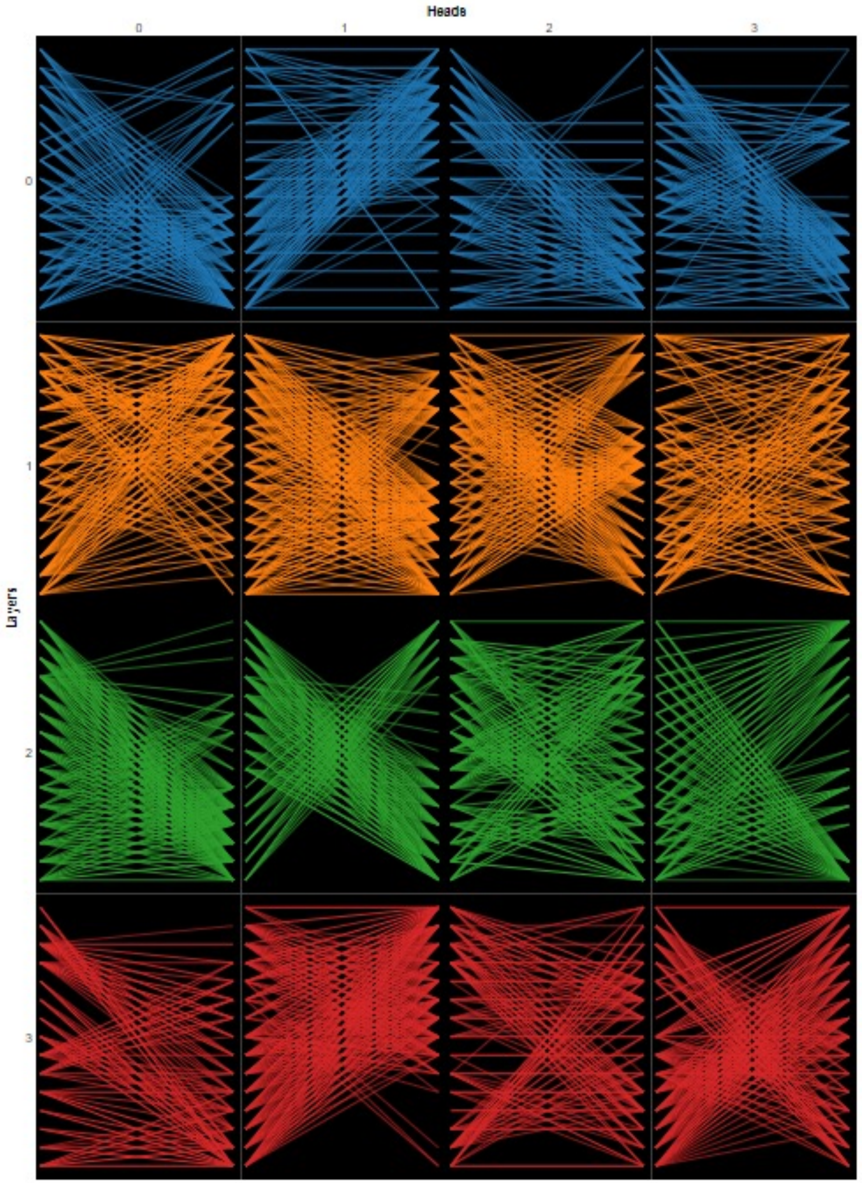

Figure 14. Attention map trained by data before shuffling

Attention Map Evaluation

	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 3. Existence of properties in attention map (after shuffling)

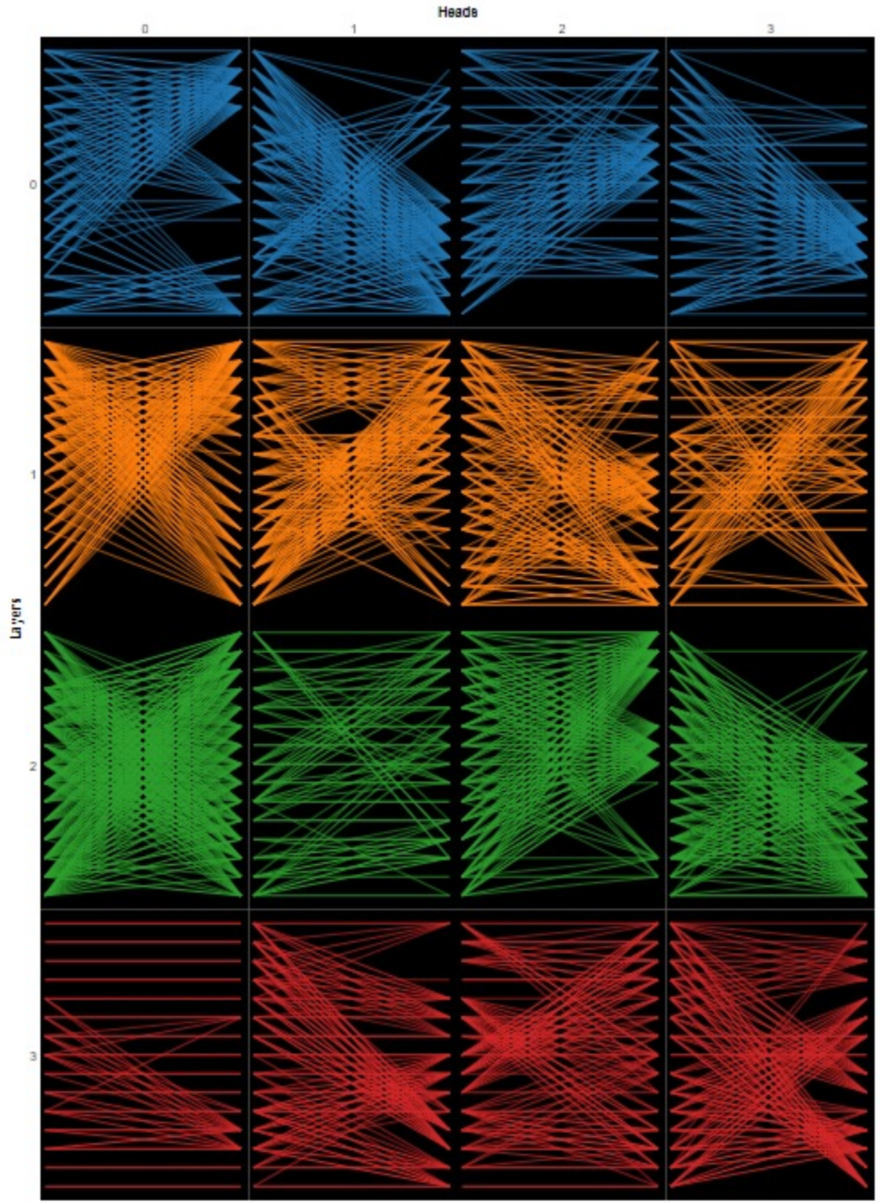

Figure 15. Attention map trained by data after shuffling

Interpretability
Contribution of Horse Tokens to the Prediction

Contribution of Horse Tokens to the Prediction

- model generalizes some ideas in the learning process regarding horse racing prediction
- look at those ideas and conceptualize them into simple rules that assist the betting
- integrated gradient is chosen to be the tool for us to realize the input-output behavior[9] of the model

Integrated Gradient

- utilize the gradient operations to compute the integrated gradients by integrating the first-order derivatives
- the input features' attribution of the model prediction can be obtained for further analysis
- Equation for mth horse token in the input sequence x contributes to the model prediction $F(x)$ is shown below[10]

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

Contribution of Horse Tokens to the Prediction

- winner contributes positively to a large extent and most other horses contribute negatively

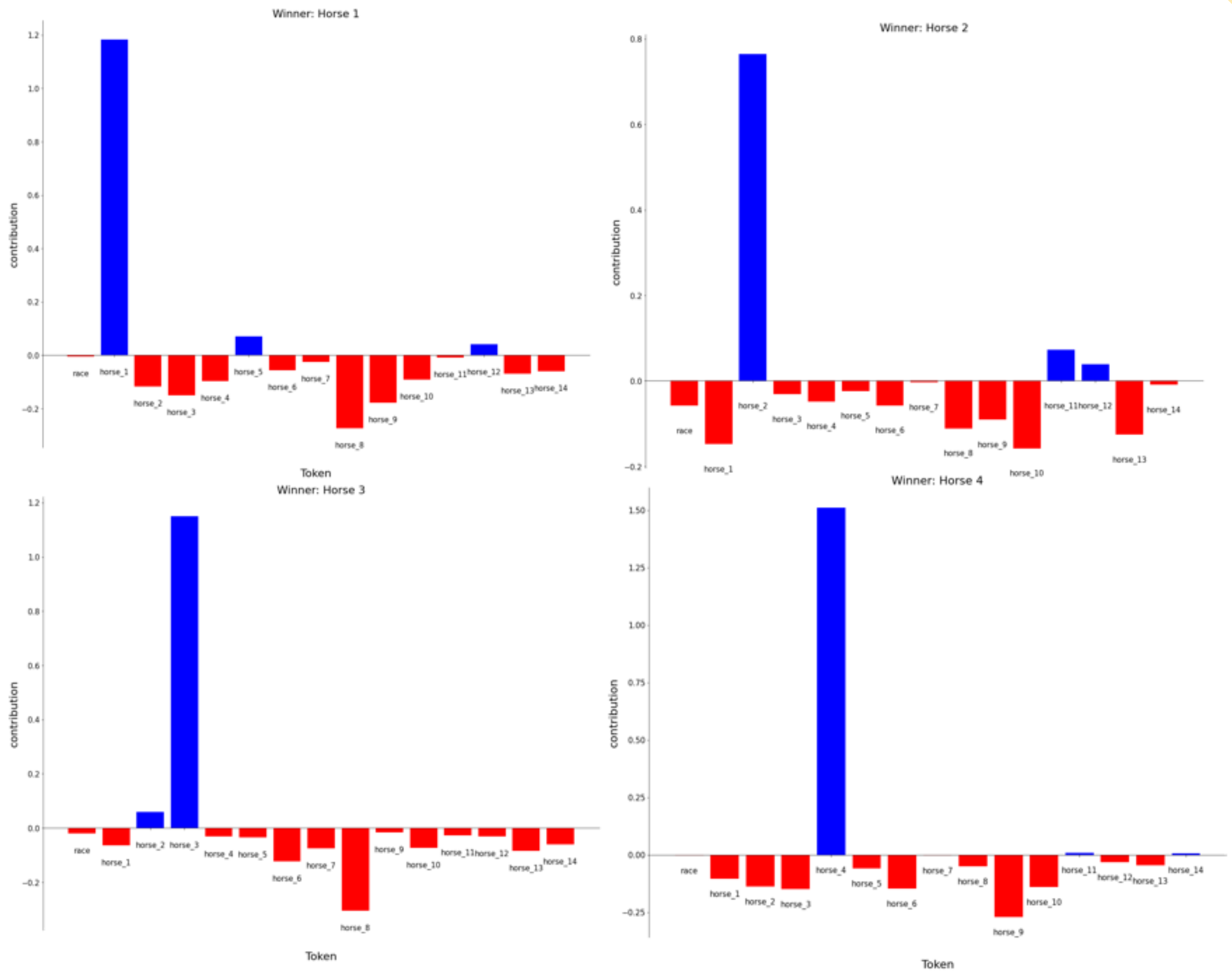

Figure 16. Contributions of horse tokens when horses 1 – 4 are winners

Contribution of Horse Tokens to the Prediction

- horse that contributes most negatively are not the horses next to the winner but the horses further away from it

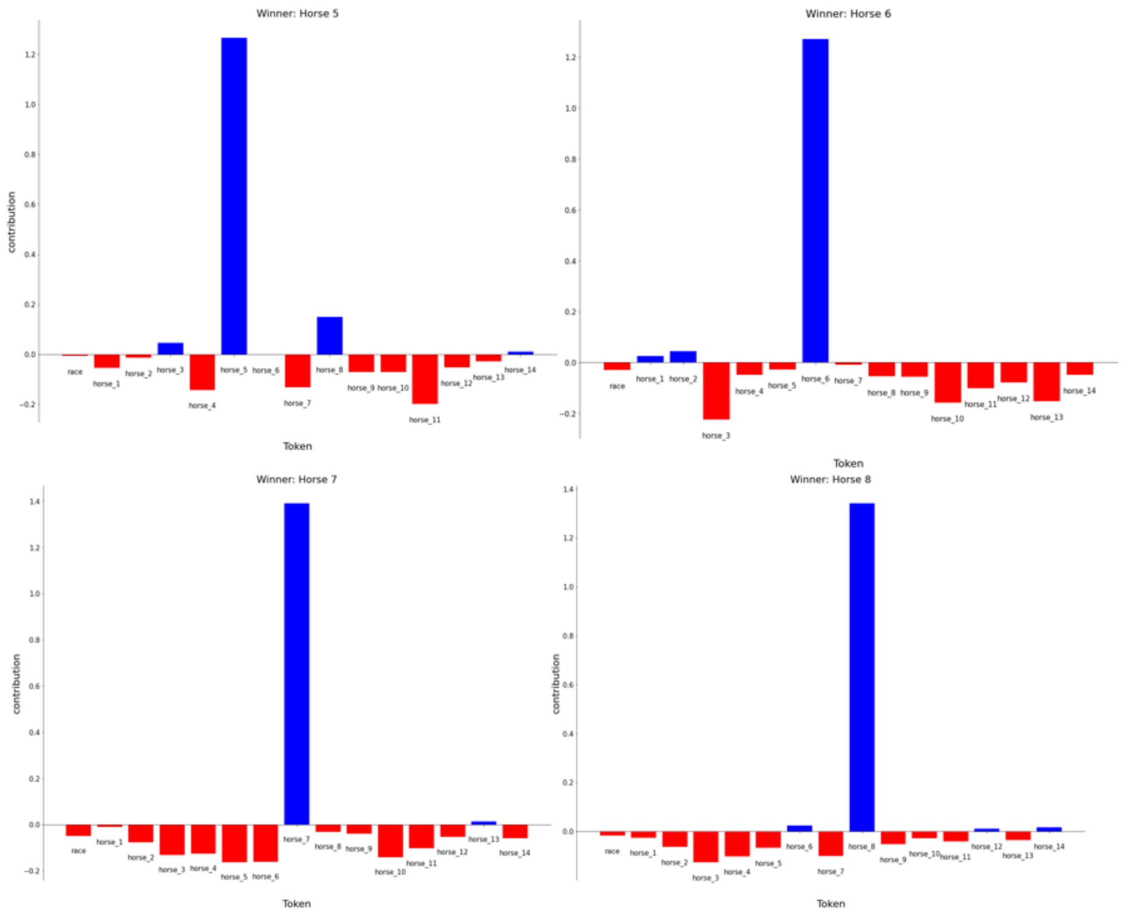

Figure 17. Contributions of horse tokens when horses 5 – 8 are winners

Contribution of Horse Tokens to the Prediction

- for winners with horse number 12 to 14, the input regarding the race information contributes a positive value

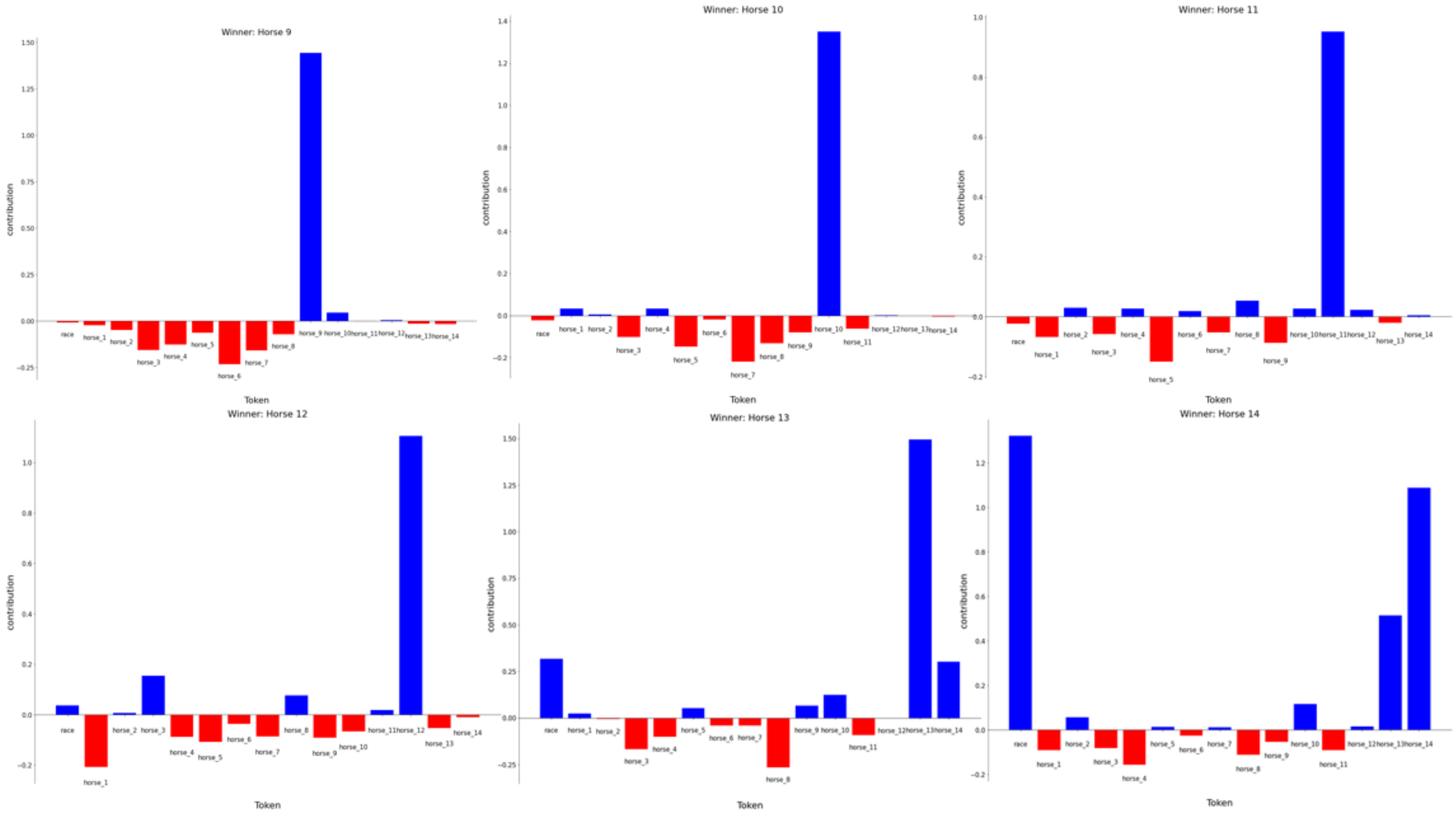

Figure 18. Contributions of horse tokens when horses 9 – 14 are winners

Simple Rules Extraction

- Rule 1
  - the negative impact of horse j will likely be increased with $|i - j|$
  - Given that you want to bet on a horse with horse number i for $1 \leq i \leq 14$
  - focus on the horses far from horse i and consider their ratings
  - If those horses' ratings are high, probability for horse i is to win is lower

Simple Rules Extraction

- Rule 2
  - race condition contributes a significant amount of positive value when the winners are horses with a number greater than 11
  - Given that you want to bet on horse i for $12 \leq i \leq 14$,
  - 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

Conclusion

Conclusion

1. Model Improvement
   - Change of Embedding method
     - Increase Accuracy
     - Maintain a steady growth of net profit
2. Investigation about Interpretability of the model
   - Assessment to Model Capability
     - Number of participant
     - Most popular horse
   - Performance Change due to data Shuffling
     - Information lost
     - Poor behaviour of attention heads
   - Horse Token Contribution to the Prediction
     - Simple rules extraction

Thank you!

References

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

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

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

[4] 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.

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

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

References

[7] 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

[8] 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.

[9] 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 (EMNLPIJCNLP), 2019.

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