COMPARISON OF DIFFERENT METHODS FOR HORSE RACING PREDICTION

Abstract
Comparison of different models on predicting the finishing time of horses

CHAN, Siu Chung
1155157657

Introduction

Horse Racing is a traditional sport that people love to watch and gamble on. Hong Kong, one of the most well-known and leading places for international horse racing competition, has a mature record and gambling system. Hence, numerous works has been done and research on this problem. However, as there are so many possible ways to approach to this question, it is hard to determine whether the performance of a model is good or not.

In this report, we will compare different model on the ability to predict the horse racing by finishing time.

Related Work

Thanks to the invention of ANN [1], we can use it either on small models like a fully-connected model or larger, more complicated models.

LSTM, long-short term memory [2] has make it possible for time-series data to record the information in the past and calculate how it affect by back-propagation.

XGBoost [3], an efficient implementation of gradient boosting is invented and it is built on decision trees [4] in sequential manner. It improves from its prediction error and make the model better each iteration.

Adam [5], A Method for Stochastic Optimization made the fast training time and high accuracy combining best properties of the AdaGrad [6] and RMSProp [7]

Feature selection is also important when building the pipeline and so this paper provide insights on how feature selection affect models [8]

Also I have studied on different algorithms [9]and get the inspiration on using what kinds of loss function throughout the report [10]

Data

As we focuses on the local horse racing data, we need a data source that is reliable. There are data source on the market that is paid by month or paid by usage. However, it may not be able to customise for what is needed. So, in the project, data is scarped from the website.

First we identify all the horses that are currently racing in the season from the [HKJC website] [11].

English Name	Former Name	Sire	Dam	Import Type	Country Of Origin	Age	Foaling Date
A AMERIC TE SPECSO	----	Per Incanto	Island Time	PPG	NZ	5	21/09/2018
ABSOLUTE SUNSHINE	Shuriken	Tosen Stardom	Petroica	PP	AUS	4	06/11/2019
ACA POWER	Keen Power	Zoffany	Lavender Bay	PP	AUS	7	10/09/2016
ACCOLADE START	Wahaaj	Gregorian	La Belle Maison	PP	IRE	4	17/03/2020

Figure 1: **Racing Information (Local)**

In Figure 1, There are the column “English Name”, “Former Name”, “Sire”, “Dam”, “Import Type”, “Country of Origin”, “Age” and “Foaling Date”. In this page, we are interested to every horses’ detail. So we need to get all link that relate to each horse. By that, we scrape all the URL in this table using BeautifulSoup (bs4), one of the powerful library of Python for web scraping.

Race Index	Pla.	Date	RC/Track/ Course	Dist.	G	Race Class	Dr.	Rtg.	Trainer	Jockey	LBW	Win Odds	Act. Wt.	Running Position	Finish Time	Declar. Horse Wt.	Gear	Video Replay
23/24 Season																		
558	03	03/04/24	ST / AWT	1650	GD	3	9	81	P F Yiu	H Bowman	SH	3.9	131	9 9 10 3	1.37.78	1142	B	
467	07	03/03/24	ST / AWT	1650	GD	2	6	81	P F Yiu	A Hamelin	2-1/4	2.9	118	9 9 10 7	1.40.00	1171	B	
436	01	18/02/24	ST / AWT	1650	GD	3	9	75	P F Yiu	A Hamelin	3/4	5.2	131	7 6 7 1	1.38.67	1164	B	
368	02	24/01/24	ST / AWT	1650	GD	3	3	75	P F Yiu	H Bowman	1-3/4	4.5	130	10 10 11 2	1.38.35	1167	B	
277	01	23/12/23	ST / AWT	1650	GD	3	3	70	P F Yiu	J McDonald	N	3.6	122	10 8 9 1	1.38.61	1153	B	
219	04	03/12/23	ST / AWT	1650	GD	3	11	71	P F Yiu	J McDonald	4	3.5	127	9 9 7 4	1.38.65	1134	B	
198	04	26/11/23	ST / AWT	1200	GD	3	4	72	P F Yiu	J McDonald	2-3/4	3.9	129	7 7 4	1.09.29	1142	CP-/B2	
116	03	25/10/23	ST / AWT	1650	GD	3	8	72	P F Yiu	K Teetan	1-3/4	2.3	129	3 3 3 3	1.38.58	1134	CP	
22/23 Season																		
731	02	10/06/23	ST / AWT	1800	WF	3	5	70	P F Yiu	K Teetan	SH	2.8	122	4 5 5 6 2	1.46.84	1114	CP	
652	02	10/05/23	ST / AWT	1650	GD	3	11	70	P F Yiu	K Teetan	4	4.8	129	9 6 6 2	1.37.79	1108	CP	
602	03	23/04/23	ST / AWT	1200	GD	3	4	70	P F Yiu	K Teetan	3-1/2	3.3	121	3 3 3	1.08.63	1107	CP	
472	01	05/03/23	ST / AWT	1200	GD	3	9	62	P F Yiu	K Teetan	N	1.8	115	5 3 1	1.07.79	1133	CP	
411	01	12/02/23	ST / AWT	1200	GD	4	10	50	P F Yiu	K Teetan	6-1/4	4.2	125	3 3 1	1.08.90	1132	CP	
352	01	21/01/23	ST / AWT	1200	GD	4	5	44	P F Yiu	K Teetan	1-1/2	8.4	119	4 4 1	1.08.43	1135	CP	
233	09	11/12/22	ST / Turf / "A"	1400	G	4	2	46	P F Yiu	K Teetan	8	8.4	121	5 6 5 9	1.23.46	1136	B-/CP2	
164	06	12/11/22	ST / Turf / "A+3"	1400	GF	4	1	48	P F Yiu	J McNeil	3-1/2	8.6	128	4 3 4 6	1.22.11	1130	CP-/B1	
099	05	16/10/22	ST / Turf / "A+3"	1400	GF	4	13	50	P F Yiu	K C Leung	3	13	125	14 14 11 5	1.22.17	1126	CP	
006	08	11/09/22	ST / Turf / "A"	1200	G	4	11	50	P F Yiu	K C Leung	2-1/2	8.8	125	4 4 8	1.11.12	1126	CP	
21/22 Season																		
789	08	01/07/22	ST / Turf / "C"	1200	GY	4	5	52	P F Yiu	C Y Ho	4-1/4	6.2	128	10 8 8	1.13.08	1133	CP/TT-	
743	05	12/06/22	ST / Turf / "C+3"	1000	G	4	8	52	P F Yiu	K C Leung	3-1/4	42	125	7 7 5	0.56.01	1123	CP1/TT1	

Figure 2: **Horse Form Records (All)**

For each horse, there is a table called Horse Form Records (All) with the structure like Figure 2. The columns are “Race Index”, “Pla”, “Date”, “RC/Track/Course”, “Dist.”, “G”, “Race Class”, “Dr.”, “Rtg.”, “Trainer, Jockey”, “LBW”, “Win Odds”, “Act.Wt.”, “Running Position”, “Finish Time”, “Declar. Horse Wt.”, “Gear” and “Video Replay”. We scrape all the columns in the table using bs4. It is then stored in a .csv file using Pandas DataFrame.

```
       Race_Index  Place      Date  RC/Track/Course Distance  Going  \
count       15210  15210     15210            15210    15210  15210
unique        838     24       496               17       12     11
top           449     01  12/02/24  ST / Turf / "A"     1200      G
freq           37   1599       136             1912     6180  11013

       Race_Class   Draw Rating   Trainer    Jockey    LBW Win_Odds  \
count       15210  15210  15210     15210     15210  15210    15210
unique         15     15    123        27        65    155      353
top             4      5     52  A S Cruz  Z Purton  1-1/4       10
freq         6882   1294   1240      1031      1229    606      575

       Actual_Weight Running_Position Finish_Time Declared_Horse_Weight  \
count          15210            15210       15210                 15210
unique            30             4983        3104                   375
top              126               --          --                    --
freq            1124              246         246                   239

         Gear Video_Replay Video_Replay_2
count   15210        15210          14971
unique    651            2              1
top        --
freq     2920        14971          14971
```

Figure 3: **Example DataFrame after scarped**

The DataFrame column is almost the same as the table shown in figure 2, except that some table might have a column called Video_Replay_2. Each row represents a horse data in Figure 1.

After scraping the data and store it in the csv, we can use the csv to do pre-processing to meet the needs for different model. There are some common pre-processing steps that we can apply to data before the data is further pre-processed before training.

There are a few columns that we do not concern much as they weight less in contributing to the prediction [8], they are “Trainer”, “Jockey”, “LBW”, “Running_Position”, “Gear”, “Video_Replay”, “Video_Replay_2”. They have negligible impact on the model accuracy so it is better to drop it from the column in order to reduce complexity. We should also separate “RC/Track/Course” as it is one column in the original table. Next, we parse the time to milliseconds. For “Place”, “Win_Odds”, “Draw”, “Rating”, “Actual_Weight”, “Declared_Horse_Weight”, we have to parse it into numerical column, output NaN if it cannot be parsed. Also strip the “Course” column. Some rows contains “--” as the value so it is meaningless, we will consider dropping that as well.

Approach

Since this project compare the ability of different models on predicting the finishing time, data are tested on multiple models. One-Hot Encoding, minmax scaling and standard scaling are applied to the data accordingly For all models, train-test split is used at a 7:3 scale. There are 3 models, fully-connected network [1], LSTM [2] and XGboost [3]

Fully-connected network

First, we have to drop the unrelated columns such as 'Race Index' and 'Date' that are irrelevant. After that we do One-Hot Encoding, minmax scaling and standard scaling to the data accordingly. The final data has 28 features.

Then build the model architecture:

```
HorseRacingModel(
  (fc1): Linear(in_features=28, out_features=2048, bias=True)
  (relu1): ReLU()
  (fc2): Linear(in_features=2048, out_features=2048, bias=True)
  (relu2): ReLU()
  (fc3): Linear(in_features=2048, out_features=1, bias=True)
)
```

Figure 4: **Model Architecture of Fully-connected network**

There are 3 layers, input layer with 28 input features, hidden layer with 2048 node and output layer with 1 feature which is the finishing time (ms). This model predicts one horse at a time.

LSTM

LSTM is also being studied on how it predict the finishing time of each horse given an array of horses. Each index of the array represents the features of a horse. The input features is also 28 in this model.

```
LSTMModel(
  (lstm): LSTM(28, 1024, batch_first=True, bidirectional=True)
)
```

Figure 5: **Model Architecture of LSTM Model**

The LSTM used consist of one LSTM block which has 1024 hidden layer. Since it should consider the effect of horses in different position so it is implemented in bidirectional.

XGBoost

XGBoost is also trained using xgb library,

```
# Initialize the model
xgb_reg = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=51, learning_rate=0.1, seed=42)
```

Figure 6: **Implementation of XGBoost Model**

Here we Initialize the model with squared error and regression task since it is going to predict the finishing time of horses. We will predict one horse at a time using this model.

Experiments

Fully-connected network

```
# Model Training
input_size =  28
hidden_size =  2048
output_size =  1

model = HorseRacingModel(input_size, hidden_size, output_size).to(device)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

num_epochs = 500
```

Figure 7: **Hyper-parameters of Fully-connected network**

For hyper-parameter, we have the loss function of MSE Loss and optimizer for Adam with lr=0.001. We trained the model for 500 epochs.

The result, stopped at plateau after some trials.

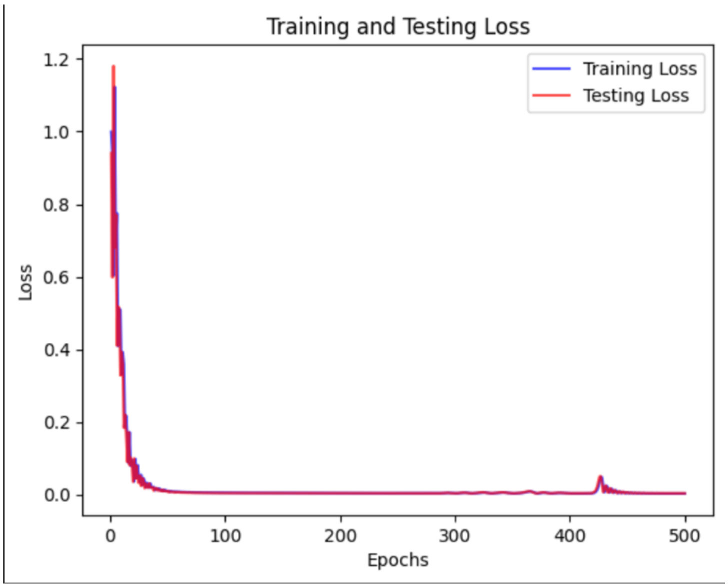

Figure 8: **Loss curve for fully-connected network**

The testing loss for this model is 0.0035405, which by multiplying the variance of the finishing_time column, is equivalent to 1065 ms, slightly more than a second to predict a horse finishing time with its features.

LSTM

```
input_size = 28
hidden_size = 1024
num_layers = 1
output_size = 1

model = LSTMModel(input_size, hidden_size, num_layers, output_size).to(device)

criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)

num_epochs = 500
scheduler = CosineAnnealingLR(optimizer, num_epochs)
```

Figure 9: **Hyper-parameters of LSTM**

For hyper-parameter, we have the loss function of MSE Loss and optimizer for Adam with lr=0.01. We trained the model for 500 epochs with a CosineAnnealingLR.

The result, stopped at plateau after some trials.

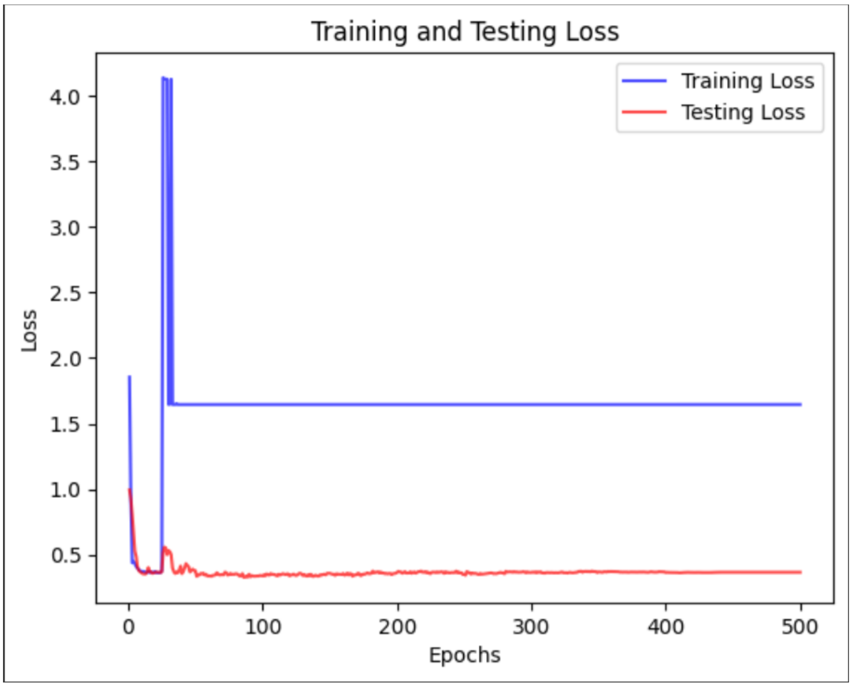

Figure 10: **Loss curve for LSTM**

The test loss of this model is 0.3653 which is equivalent to 10827 ms error. Around 10 second error. The result of the LSTM model is not satisfactory due to a number of reasons: Data size, inconsistent input sequence length and the data is not time series data.

The data size is very small since it only contains recent years data. Moreover, some horses in some of the race has already retired so I cannot scrape the horse info. This leads to small data size. Furthermore, the horse number of each race is different so it is very hard for the model to know which horse should refer to which output slot in the output.

XGBoost

```
# Initialize the model
xgb_reg = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=51, learning_rate=0.1, seed=42)

# Train the regressor with evaluation metrics
xgb_reg.fit(X_train, y_train,
            eval_set=[(X_train, y_train), (X_test, y_test)],
            eval_metric='rmse',
            verbose=True)
```

Figure 11: **Hyper-parameters of XGBoost**

For hyper-parmeters, the number of estimators is set to 51 and learning rate is 0.1. Squared Error is used in this case.

Since the evaluation metric is rmse, so the mse loss is the square of rmse. The loss graph is as follows,

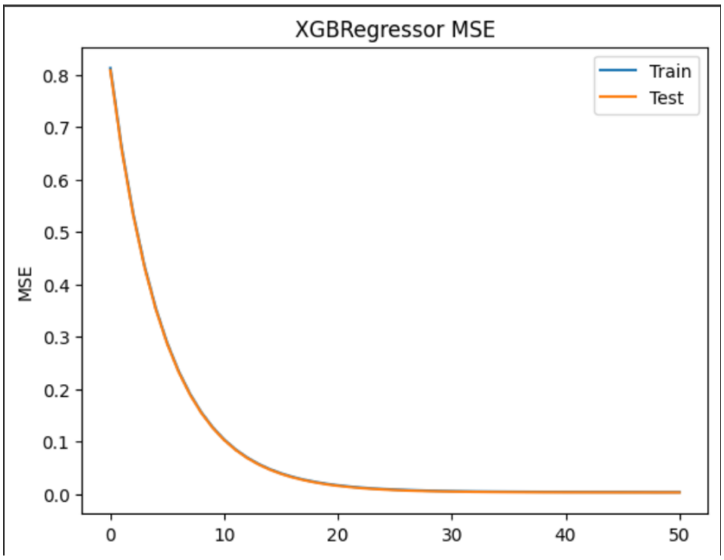

Figure 12: **Loss curve for XGBoost**

The MSE loss for XGBoost is 0.002477. Which is equivalent to 891.5966 ms.

Conclusion

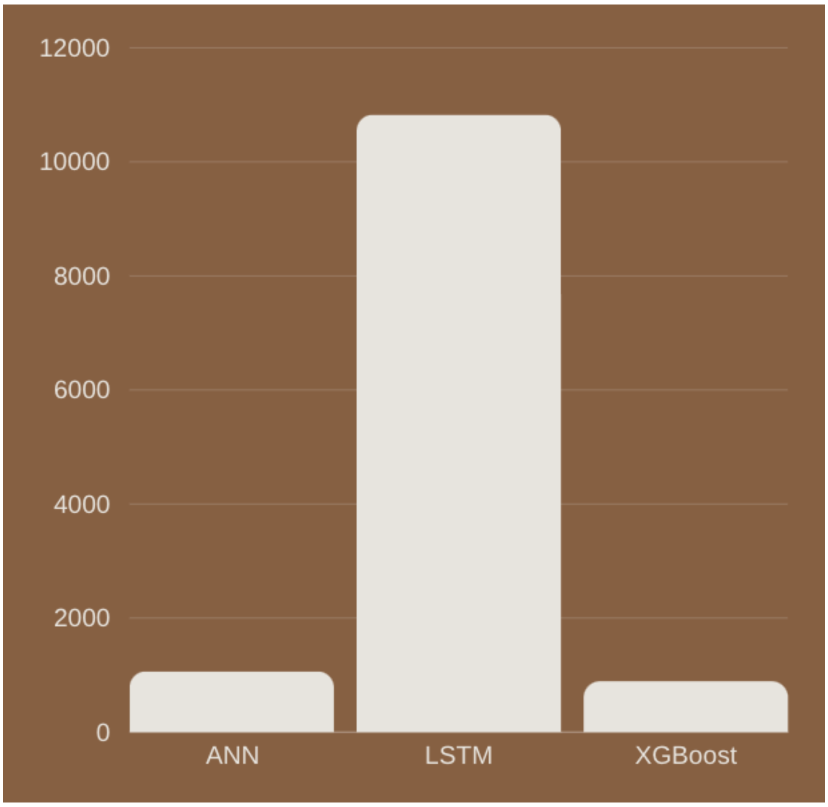

Figure 13**: comparison of model loss in ms**

In conclusion, if we want to predict the finishing time of each horse, the prediction of horse one by one which ANN and XGBoost uses, is better than predicting them by batch like LSTM. It is because there is limitation on the size of dataset and different sequence length between race.

Among three models, XGBoost performs the best because it is exceptionally good at dealing with tabular data. The weak learner in XGBoost can fix the previous models error so it converge faster, unlike ANN approach, learn from the original data.

ANN is good for general data like the tabular one, but not as good as XGBoost as limited by its framework, it needs much more epoch and training time to reach more satisfactory result.

Although LSTM can learn from the previous data, it is hard for it to learn something that is non time series related because the output sequence may not have relationship.

Reference

[1] McCulloch, W. S., & Pitts, W., “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics,* vol. 5(4), pp. 115-133, 1943.

[2] Hochreiter, S., & Schmidhuber, J"urgen., “Long short-term memory,” *Neural Computation,* vol. 9(8), p. 1735–1780, 1997.

[3] Chen, T., & Guestrin, C., “XGBoost: A Scalable Tree Boosting System,” *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 785-794, 2016.

[4] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., … others., “ Top 10 algorithms in data mining,” *Knowledge and Information Systems,* vol. 14(1), pp. 1-37, 2008.

[5] Diederik P. Kingma, Jimmy Ba, “Adam: A Method for Stochastic Optimization,” in *3rd International Conference for Learning Representations, San Diego, 2015*, San Diego, 2015.

[6] Duchi, J., Hazan, E., & Singer, Y., “Adaptive subgradient methods for online learning and stochastic optimization.,” *Journal of Machine Learning Research,* vol. 12, pp. 2121-2159, 2011.

[7] G. Hinton, “Lecture 6e: RMSProp - Divide the gradient by a running average of its recent magnitude.,” 2012. [Online].

[8] Gupta, M. ., & Singh , L. ., “Horse Race Results Prediction Using Machine Learning Algorithms With Feature Selection,” *International Journal of Intelligent Systems and Applications in Engineering,* vol. 12(2s), pp. 132-139, 2023.

[9] Williams, Janett and Li, Yan, “A case study using neural networks algorithms: horse racing predictions in Jamaica,” in *ICAI 2008: International Conference on Artificial Intelligence*, Las Vegas, NV. USA, 2008.

[10] Davoodi, E., & Khanteymoori, A. R., “Horse racing prediction using artificial neural networks.,” *Recent Advances in Neural Networks, Fuzzy Systems & Evolutionary Computing,* pp. 155-160, 2010.

[11] “HKJC,” [Online]. Available: https://racing.hkjc.com/racing/information/English/Horse/HorseFormerName.aspx. [Accessed 18 4 2024].