Horse racing result prediction using Extreme Gradient Boosting

CHENG Wing Ryan 1155102964

Contents

Introduction

Predict the outcome of horse racing by machine learning is always an interesting topic, especially in Hong Kong. William Benter and Robert Moore prove its power a few decades ago. Thanks to the Pari-mutuel Local Pools system and the relatively complete data from HKJC. With Extreme Gradient Boosting (XGBoost), one of the most favourable machine learning method in Kaggle, letting profit consistently with machine learning algorithm from horse racing feasible.

Model introduction

Due to the limited amount of data and the computer limit, rather than using neural network, boosting should be a better choice since it relies on fewer data and computational power. There are several well-known boosting algorithms such as random forest and gradient boosting machine. Base on those boosting algorithms, Extreme Gradient Boosting (XGBoost) was invented. It is well known for its performance on training speed and the computational result.

Model concept explanation

As mentioned before, XGBoost is a type of boosting method. It is an ensemble learning method composed of several base learners. Mathematically speaking, we first have

$\mathrm{F} = \{\mathrm{f}_1, \mathrm{f}_2, \dots, \mathrm{f}_{n-1}, \mathrm{f}_n\}$ set of base learners.

The aim to find $\hat{y} = \sum_{t=1}^{m} f_t(x_i)$.The following figure represents how it works.

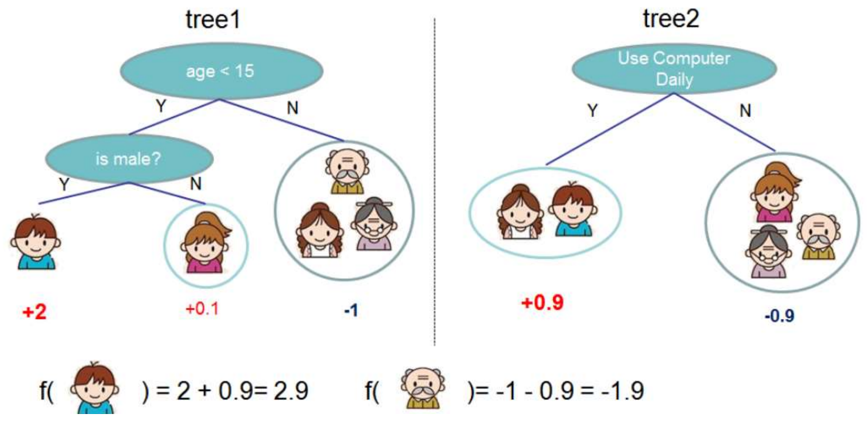


FIGURE 1(TIANQI CHEN, 2016)

Let our sample be $O = \{x_1, x_2, \dots, x_{n-1}, x_n\}$

The objection function at iteration t is $L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}^{t-1} + f_t(x_i)\right) + \Omega(f_t), \Omega(\mathrm{f}) = \gamma\mathrm{T} + \frac{1}{2}\lambda\left|\left|w\right|\right|^2, T = leaf\ size, w = weight\ of\ leaf$, where the $l\left(y_i, \hat{y}^{t-1} + f_t(x_i)\right)$ part is the loss function part, the $\Omega(f_t)$ part is the regularization part and y_i is considered as the label from the training dataset.

By applying a second-order Taylor Series into L^t, we finally obtained $\hat{L}^{(t)} = \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t)$.

One reason that XGBoost is faster than other boosting algorithm is that it used Taylor Series, thus reducing the computational power. And the second reason is it transforms the function to the Euclidean domain to apply the tradition method. As we can see, $L^{(t)}$ is a nested function, and the authors mentioned that “cannot be optimized using traditional optimization methods in Euclidean space.” (Tianqi Chen, 2016) With a bunch of calculation, we finally obtain

$\hat{L}^t(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i+\lambda} + \gamma T, I_j = \{i|q(x_i = j)\}$, where $\left(\sum_{i\in I_j} g_i\right)^2$ is the set that mapped to leaf j.

The below figure may explain it more clearly. D1, D2 and D3 both represent each characteristic of box 1, box 2 and box 3. XGBoost figured the Box 4 out by ensemble all characteristics together and weight them.

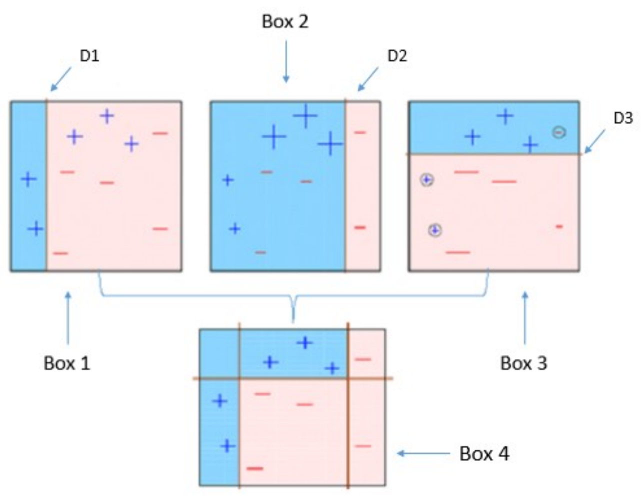


FIGURE 2(DATACAMP)

Advantage of XGBoost

It has a built-in Lasso Regression and Ridge Regression regularization to prevent the model from overfitting and It can implement parallel processing and even take advantage of GPU. Since it is coded in C and it takes advantage from Taylor Series, thus it has a fast training speed. And finally it can deal with the missing value, although deal with the missing during the missing data process may give us a better clue of how our data works.

Disadvantage of XGBoost

It does not perform well in terms of auto-regressive data compare to time independent data. Furthermore, it could not work with labelled data, a conversion must be made before using the model. And finally, it is a 'black box' model, we cannot interpret it directly.

Data description

All data is collected from the HKJC website. Please refer to the *appendix*.

Dependent variable: place of 'current' game

Since XGBoost cannot deal with labelled data, we first use label encoding to convert all label. Please note that one-hot encoding will not be considered due to the limit of computational power. Furthermore, for each race, the record of horses which disqualified by accident will be removed.

Feature engineering

Some data mentioned above could not obtain before the race, such as the racing time. Thus, we can only use past data. However, we can still obtain some data at the 'current' race such as the track condition of the race, or the weight of the horse, and most importantly, the weight of the horse. Result tells us the odds are the most important variables to predict the result.

Besides from using all the raw data, some feature engineering has been done to get some unique data to enhance the prediction result.

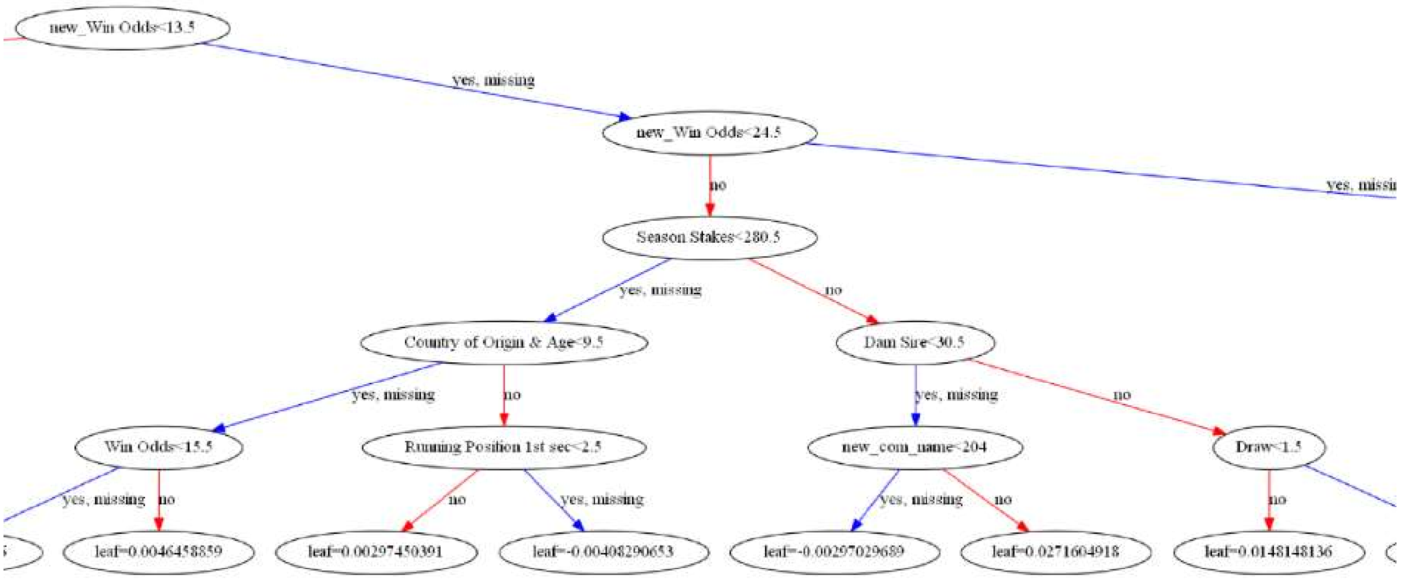
Variable name	Description	Type	Values
Actual Wt. diff	The difference of weight that the horse has to burden in that particular game and the previous game	continuous	int
Declar. Horse Wt. diff	The difference of weight that the horse in that particular game and the previous game	continuous	int

Data splitting

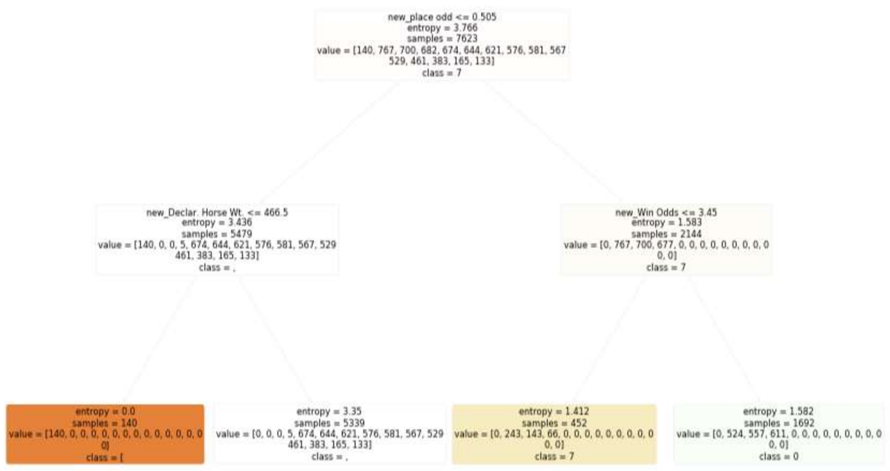
The whole set of data is divided into train and test groups (70%/30%) with an order. Since the data relies on the previous result, shuffle them may lead to a 'cheating' problem.

Model explanation

Since XGBoost is a Blackbox algorithm, we cannot explain it directly. However, we can take advantage of its structure and the variable importance plot to interpret it. The below figure is one of the decision trees under XGBoost, it is an only a partition of the original tree.



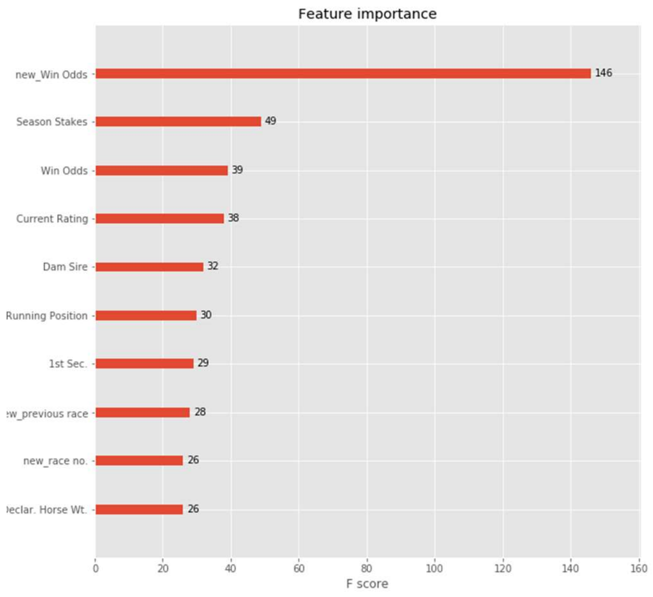
Let us construct a real decision tree to see how XGBoost make decision.


new_place odd <= 0.505
entropy = 3.766
samples = 7623
value = [140, 767, 700, 682, 674, 644, 621, 576, 581, 567
529, 461, 383, 165, 133]
class = 7
new_Declar. Horse Wt. <= 466.5
entropy = 3.436
samples = 5479
value = [140, 0, 0, 5, 674, 644, 621, 576, 581, 567, 529
461, 383, 165, 133]
class = ,
new_Win Odds <= 3.45
entropy = 1.583
samples = 2144
value = [0, 767, 700, 677, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0]
class = 7
entropy = 0.0
samples = 140
value = [140, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0]
class = [
entropy = 3.35
samples = 5339
value = [0, 0, 0, 5, 674, 644, 621, 576, 581, 567, 529
461, 383, 165, 133]
class = ,
entropy = 1.412
samples = 452
value = [0, 243, 143, 66, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0]
class = 7
entropy = 1.582
samples = 1692
value = [0, 524, 557, 611, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0]
class = 0

As we can see, decision doing pretty poor, this may cause by the depth of the tree is very shallow. So, lets look at the more powerful model, XGBoost.

Ideal case model

the below figure is the variable importance plot.

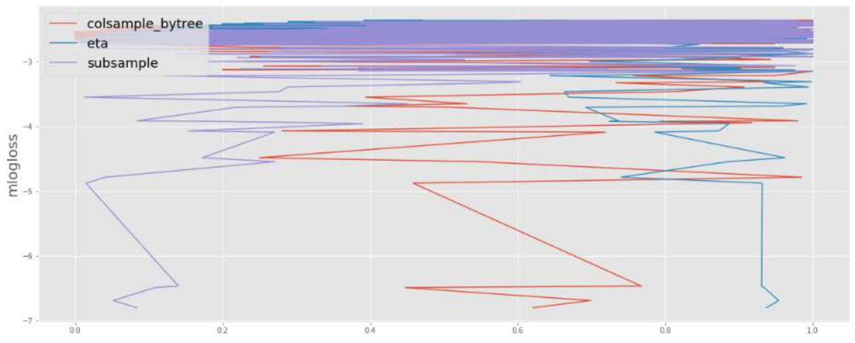


As we can see, the importance of win odds of ‘current’ game is dominating, which means the accurate odds we obtain, the better the prediction result. The reason this model is called ideal case model is because it is not possible to obtain the win odds of ‘current’ game.

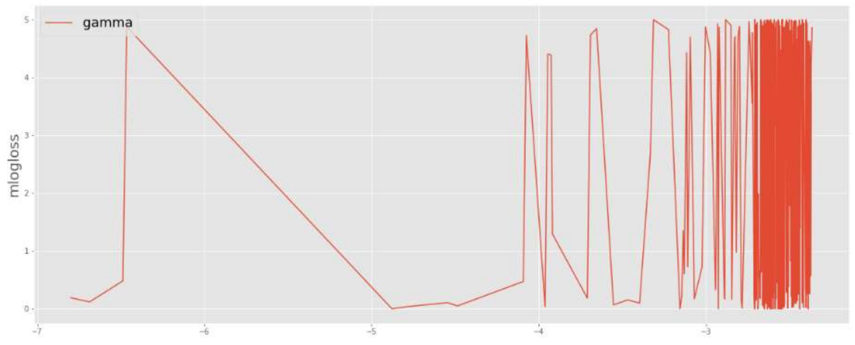
Parameter tuning

Since there are lot of hyperparameter to be tuned, using grid search may not be a wise choice. Therefore, Bayesian optimization is introduced in this report. Bayesian optimization works by finding a posterior distribution of function that fit the target function most. In this case, mlogloss is the target function that the optimization method wanted to optimize.

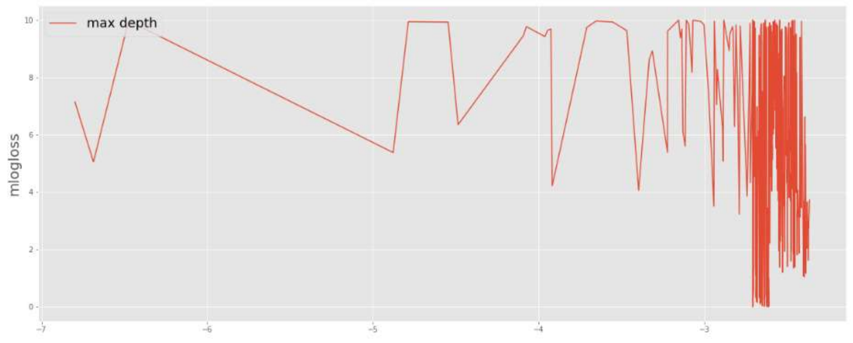
However, since the number parameters is very lager, an optimal solution still cannot be reached through Bayesian optimization. The below graph is the relationship of the parameters and the error. Where colsample_bytree means the subsample ratio of columns when constructing each tree, eta means the learning rate, subsample is how XGBoost sample the data prior to the trees, Gamma control how conservative the algorithm would be, max depth means the maximum depth of each tree and number of estimators is the number of boosting iterations.

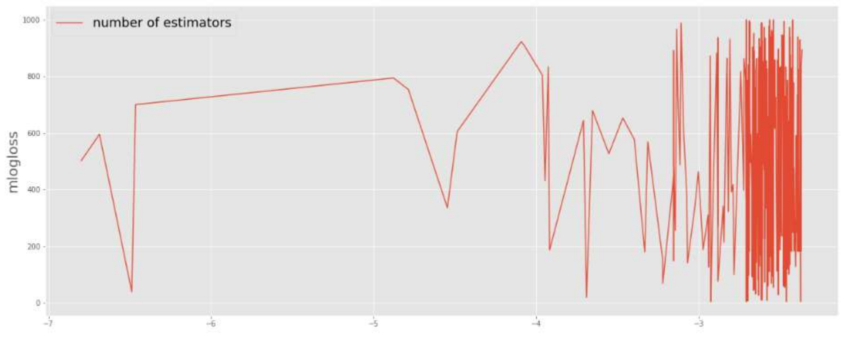


Although it doesn’t reach the optimal solution yet (due to the limitation of compactional power), we can still observe it is



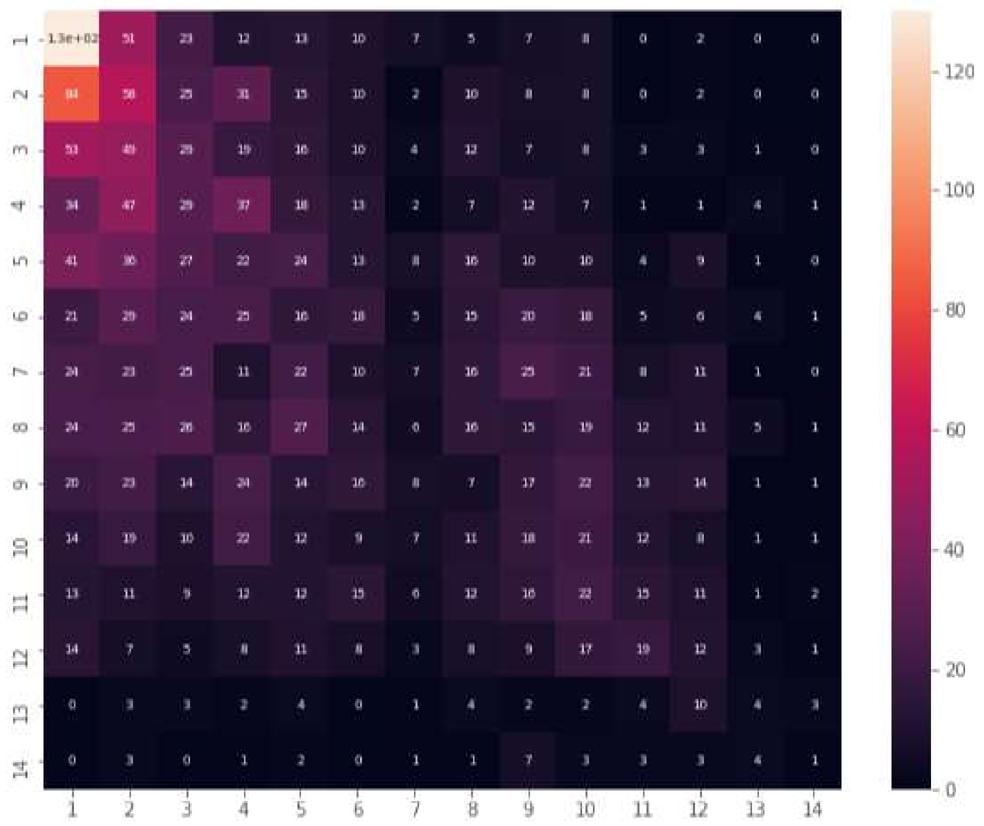
converging to an optimal point.





Accuracy of the ideal model

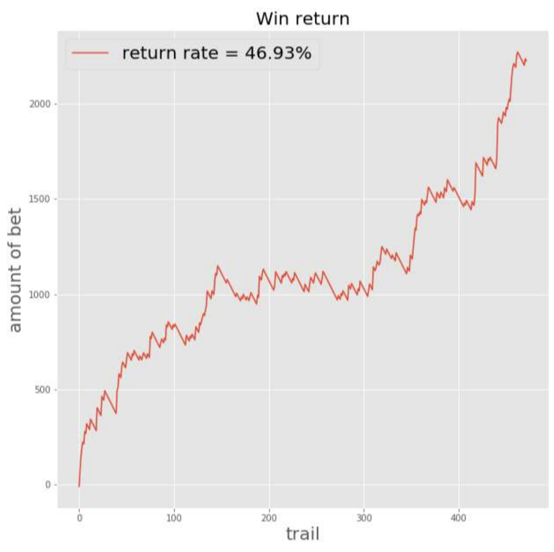
The F1 score after tuning is 0.166 and the confusion matrix is as follow

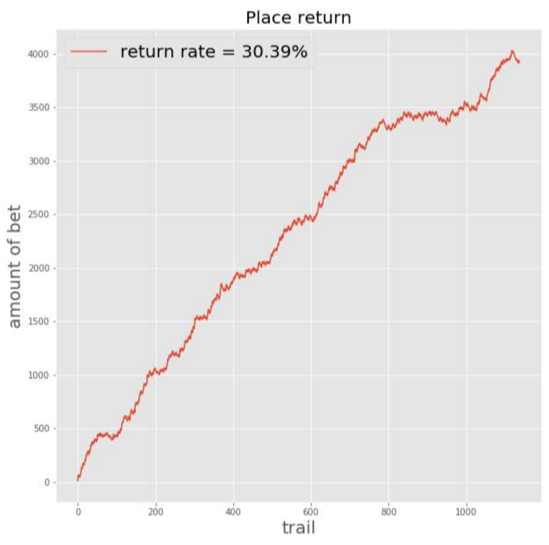


Accuracy improves from 0.145 to 0.166.

Back testing of the ideal model

The following figures shows the result of applying the model to the past data. This does not represent the actual event, it is the Monte Carlo simulation with the same amount of return. Amount of bet means the amount of betting return, each time with $10. Please note that putting all your money purely be odds will never be a wise idea, history proves that it is one of the best ways to burn all your money.





The model will only place the bet when the horses are predicted as the first 3 horses in the race.

Surprisingly, this model is already profitable. The confusion matrix can explain why this model is profitable under this relative low accuracy. When we condition on the winning horse, i.e. $P(predict\ it\ correctly|place = 1, type1\ error)$, the accuracy is 0.2908 instead of 0.166, which means this model has outstanding performance when predicting the winning horses.

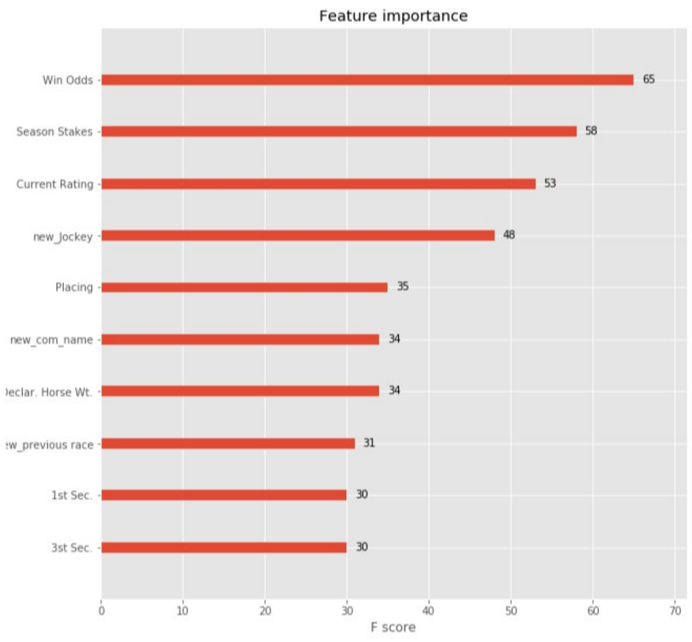
Furthermore, the impact of type 2 error will only occur implicitly, which means what we lose is the implicit profit rather than realized profit.

Limitation

The odds of the 'current' game is float, which means, in reality, it is not possible to obtain the real odds. Sadly, win odds is the most important feature in the model. the volatility of the odds before the race start can even be above 50%. In order words, it will further decrease the accuracy of the model. Thus, it motivates us to move on to the real-world case, what if we drop the most important parameter, the odds of 'current' game?

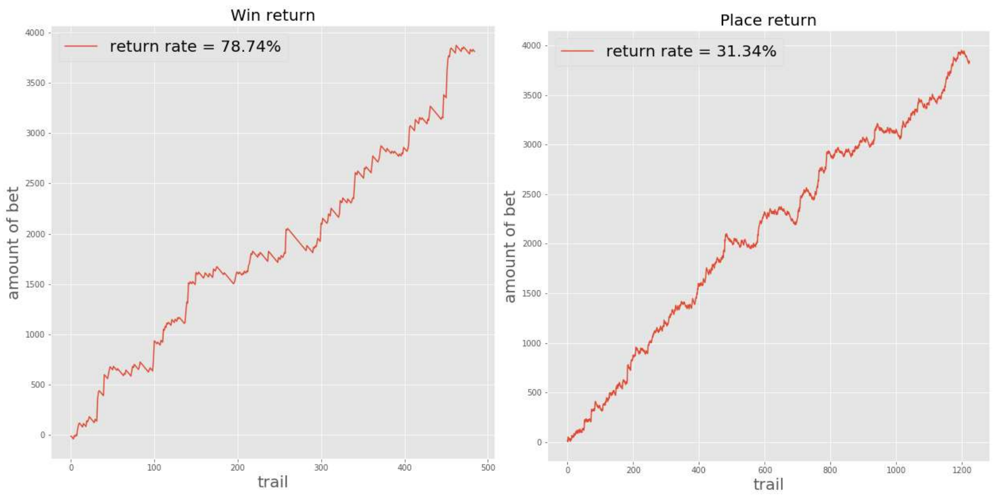
Real-world model

Using the same parameters obtained from training the real model, the F1 score is 0.1518, which is pretty encouraging. The below figure is the variable importance plot.



Back testing of the real-world model

The following figures shows the result of applying the model to the past data.

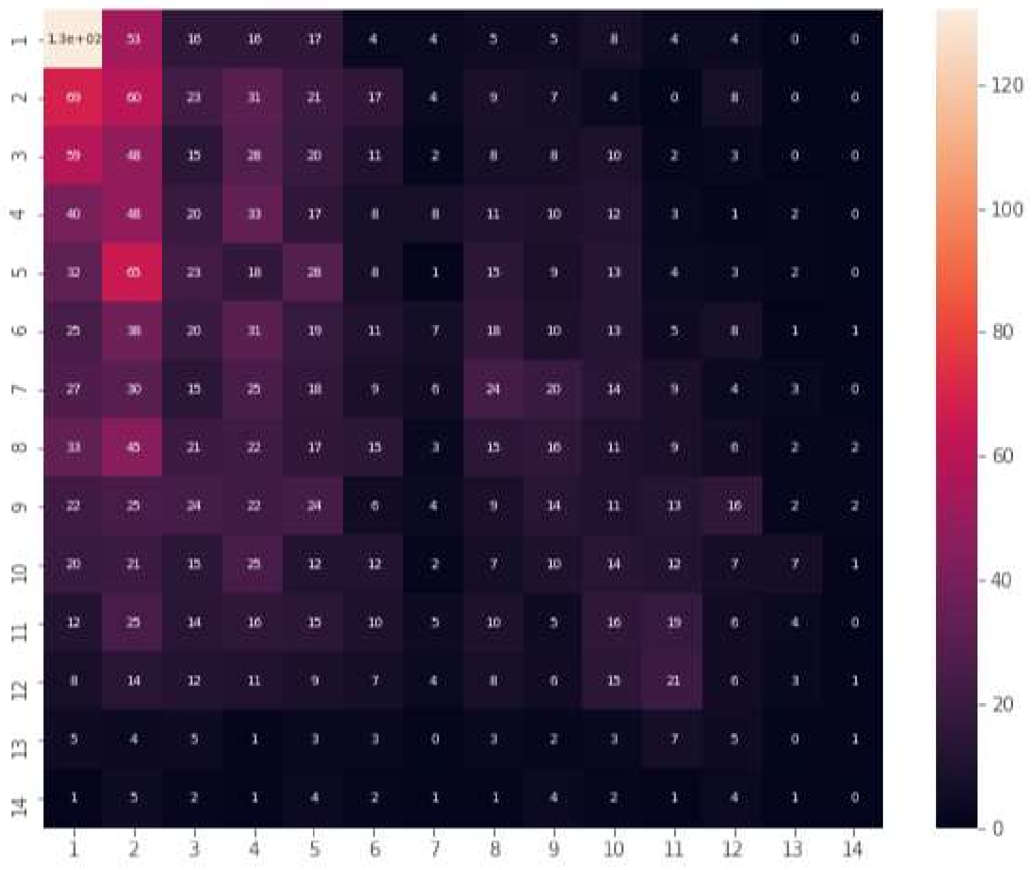


It shows a very interesting result. With a lower accuracy, the return rate increases dramatically, especially for betting on ‘win’. This may mean although the public intelligence is enough to identify the winning horse, the risk-reward ratio is not worth to bet on the high win odds horse.

Accuracy of the real-world model

The below figure is the confusion matrix of the real-world model.

$$P(predict\ it\ correctly|place = 1, type1\ error) = 0.2882$$



Potential improvement

In this project, many variables are not used wisely, much more feature engineering can be done. For example, the average performance of the last four races. The weather condition can also be considered. Besides, conditioning on some variables can be the potential approach too. The comments provided by HKJC are not used in this project too, a potential approach is applying a BERT model to see the correlation between text and horse racing result.

Rather than predicting the result directly, predict the odd using machine learning algorithm or simply a Monte-Carlo simulation first can also be the option. Although result shows as we may not need to bear the risk of doing a prediction base on a prediction.

Most importantly, a strategy on the money management is vital, Kelly criterion may be a good way to approach.

Conclusion

Overall, this project shows the power of machine learning. With those limited data and the naïve approach of feature engineering, it still provides a profitable result, which is encouraging.

Appendix

Variable name	Description	Type	Values
horse no	Horse number of that particular race	categorical	"1:99"
Horse	Horse's name	categorical	NA
Jockey	Jockey's name	categorical	NA
Trainer	Trainer's name	categorical	NA
Actual Wt.	The weight that the horse has to the burden in that particular game	continuous	int
Declar. Horse Wt.	The weight of the horse in that particular game	continuous	int
Draw	The position of the horse start from race	continuous	"1:14"
LBW	The distance of the horse from the first horse	continuous	int/SH/NA
Running Position 1st sec	Running position of the first section	categorical	"1:14"/NA
Running Position 2st sec	Running position of the second section	categorical	"1:14"/NA
Running Position 3st sec	Running position of the third section	categorical	"1:14"/NA
Running Position 4st sec	Running position of the fourth section	categorical	"1:14"/NA
Running Position 5st sec	Running position of the fifth section	categorical	"1:14"/NA
Running Position 6st sec	Running position of the sixth section	categorical	"1:14"/NA
Running Position	Final running position	categorical	"1:14"/NA
Finish Time	Finish time	continuous	float
Win Odds	Odds of winning that particular horse	continuous	float
place odd	Odds of being the first three	continuous	float

	of that particular horse		
race no.	Race number in that race day	continuous	int
Class	The strength of the horse, the better the horse, the lower the class	categorical	"1:5"
length	The length of that track	categorical	1000, 1200, 1400, 1600, 1650, 1800, 2000, 2200
Going	The condition of that track	categorical	More than 10 unique values
com_name	competition name	categorical	More than 13 unique values
course	Track	categorical	More than 7 unique values
prize	The prize of that race	continuous	float
1st Sec.	Running time of the first section	continuous	float
2st Sec.	Running time of the second section	continuous	float
3st Sec.	Running time of the third section	continuous	float
4st Sec.	Running time of the fourth section	continuous	float
5st Sec.	Running time of the fifth section	continuous	float
6st Sec.	Running time of the sixth section	continuous	float
Gear	The gear that the horse wear	categorical	more than 18 unique values
Color & Sex	The colour and sex of the horse	categorical	more than 70 unique values
Country of Origin & Age	Country, origin and age of the horse	categorical	more than 30 unique values
Current Rating	Rating of the horse	continuous	int
Dam	Mother of horse	categorical	NA
Sire	Father of horse	categorical	NA
Dam Sire	The maternal grandfather of the	categorical	NA

	horse		
Import Type	How the horse was imported	categorical	PG/PPG
Number of 123 Starts	Number of 123 Starts	continuous	NA
Number of Starts In Past Races	Number of Starts In Past Races	continuous	NA
Owner	Owner	categorical	NA
Season Stakes	The stakes of the horse in that season	continuous	NA
Start of Season Rating	the rating of the horse at the start of the season	continuous	"1:100"
Total Stakes	The total stakes of the horse	continuous	NA

Reference

(Tutorial) Learn to use XGBoost in Python. (n.d.). Retrieved from

https://www.datacamp.com/community/tutorials/xgboost-in-python

Chen, T., & Guestrin, C. (2016). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16*. doi: 10.1145/2939672.2939785

Acmayuen. (n.d.). acmayuen/HK-Horse-Racing. Retrieved from

https://github.com/acmayuen/HK-Horse-Racing/blob/master/test.Rmd

Benter, W. (2008). Computer Based Horse Race Handicapping and Wagering Systems: A Report. Efficiency of Racetrack Betting Markets World Scientific Handbook in Financial Economics Series, 183–198. doi: 10.1142/9789812819192_0019ram

Btyuhas. "Bayesian Optimization with XGBoost." Kaggle, Kaggle, 31 July 2018,

www.kaggle.com/btyuhas/bayesian-optimization-with-xgboost.

Fmfn. "Fmfn/BayesianOptimization." GitHub, 16 May 2020, github.com/fmfn/BayesianOptimization.

"XGBoost Parameters¶." XGBoost Parameters - Xgboost 1.1.0-SNAPSHOT Documentation,

xgboost.readthedocs.io/en/latest/parameter.html.