The University of Hong Kong

# COMP 3359 Artificial Intelligence Applications
# Course Project Final Report

# Horse Racing Prediction - Optimized Gambling Strategy

Yeung Tsz Chung, Chris (3035567992)
Lo Wai Ting (3035570547)

## Objective of project

We aim to build a model that gives an optimized gambling strategy for local horse races, based on the predicted winning horse, the accuracy of the prediction and the win odds. Our model should rely on reasonable features or being interpretable and it outperforms the prediction made by general publics.

## Highlight of effort

1. We construct a dataset using the past three years records of the local horse racing result from the Hong Kong Jockey Club website.
2. A profiting model using Random Forest is built, with features selection considering the variable importance and the interaction effect between the features. It can predict 40% of the winning race for the bet type ‘Place’ and successfully make more than 40000HKD in 3000 races
3. From the NN model built, we discovered that the accuracy of the prediction may not be proportional to the performance of the model, we shall consider the variance of the dataset. Hence, we consider the actual return of the model to evaluate the performance of the model.

## Data source

We found a list of horses from the HKJC website: https://racing.hkjc.com/racing/information/English/Horse/HorseFormerName.aspx, we scrap all the available horse id as a list named 'horse_info_links'.

Each horse has a unique page that contains the (1) information and background and (2) racing record of the horse with that horse id. We make use of the horse_id collected to get url for each horse and scrap the tables.

For example, the link: https://racing.hkjc.com/racing/information/English/Horse/Horse.aspx?HorseId=HK_2020_E089 contains information of the horse "A LA KING", that has the horse id = "2020_E089". The picture below gives the screen capture from the website.

A SMILE LIKE YOURS (E089)

- Form Records
- Rating/Wt/Placing
- Performance by Distance
- Trackwork Records
- Veterinary Records
- Movement Records
- Pedigree
- Other Horses

*Includes local and overseas records and earnings

| | | | |
|---|---|---|---|
| Country of Origin / Age | : NZ / 4 | Trainer | : J Size |
| Colour / Sex | : Bay / Gelding | Owner | : Vicky Tang |
| Import Type | : PPG | Current Rating | : 50 |
| Season Stakes* | : $290,000 | Start of Season Rating | : 52 |
| Total Stakes* | : $290,000 | Sire | : Per Incanto |
| No. of 1-2-3-Starts* | : 0-1-0-5 | Dam | : Diamond Start |
| No. of starts in past 10 race meetings | : 0 | Dam's Sire | : Lonhro |
| Current Stable Location (Arrival Date) | : Conghua (08/02/2021) | Same Sire | : E418 go |

Horse Form Records (Recent 3 seasons) - A SMILE LIKE YOURS

Recent 3 seasons | Other seasons | Show All

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20/21 Season | | | | | | | | | | | | | | | | | | |
| 393 | 11 | 03/02/21 | HV / Turf / "A" | 1200 | G | 4 | 10 | 52 | J Size | J Moreira | 8-1/2 | 3.8 | 125 | 5 5 11 | 1.11.55 | 1000 | -- | |
| 358 | 07 | 20/01/21 | HV / Turf / "C" | 1200 | G | 4 | 12 | 54 | J Size | J Moreira | 3-1/4 | 4 | 127 | 3 3 7 | 1.10.48 | 1006 | -- | |
| 321 | 05 | 06/01/21 | HV / Turf / "A" | 1200 | G | 4 | 7 | 54 | J Size | J Moreira | 1 | 2.8 | 128 | 2 3 5 | 1.10.64 | 1003 | -- | |
| 271 | 05 | 16/12/20 | HV / Turf / "C" | 1200 | G | 4 | 8 | 54 | J Size | J Moreira | 2-1/4 | 2.2 | 127 | 5 5 5 | 1.10.21 | 1006 | -- | |
| 215 | 02 | 25/11/20 | HV / Turf / "C+3" | 1200 | G | 4 | 6 | 52 | J Size | C Schofield | SH | 77 | 125 | 7 7 2 | 1.09.85 | 999 | -- | |

We store the information and background of the horses in ‘horse_famil.csv’ and the racing record in ‘horse_info.csv’.

We collect the win odds for place from another HKJC website: https://racing.hkjc.com/racing/information/english/Racing/LocalResults.aspx?RaceDate=2019/12/21&Racecourse=ST&RaceNo=1, we make use of the race date and race course information to compile a list of urls and get the win odds for place. The data are stored in 'place_dividend.csv'

### a. Horse_info.csv

Each row of the csv file represents the performance of a horse in a particular race.

We first drop information that cannot be obtained before the start of the race, including 'RunningPosition', 'LBW' (refer to the difference in distance of the horse with the winning horse)

Features we transform and keep:

1. Finish Time: we convert them from timeframe into milliseconds.
2. Gear: Since there are different combinations of gears wore by the horse in a race, we introduce a new column 'Number of Gear' to represent this feature.
3. One-hot encoding
    1. G: refer to the condition of the track
    2. RaceClass: Race class of the race is determined by the rating of the participated horses, usually horses in the same race are from the same ranking class
    3. RC: race course of the race
    4. Track: track of the race
    5. Course: course of the race

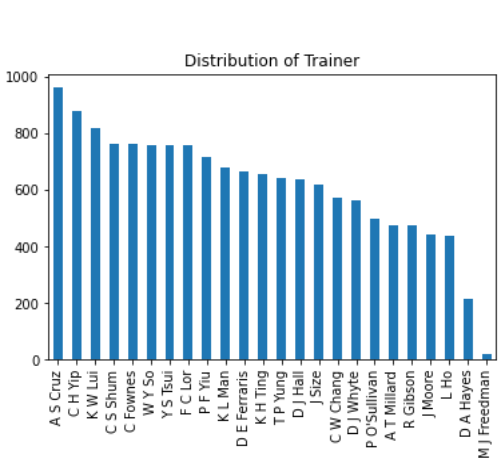
Features we keep:

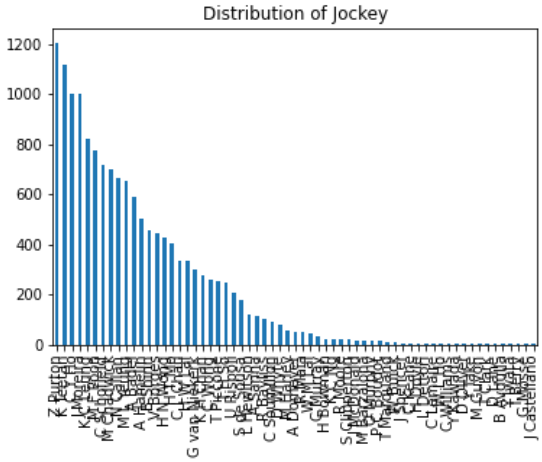
1. Pla.: refer to the place of the horse in the race
2. Dist. : refer to the distance of the race, range from 1000m to 2400m
3. Win Odds: refer to the win odds of the horse if it gets the first place
4. Declare horse weight: total weight of the horse including the weight he needs to carry
5. Draw: refers to a horse position in the starting gate. The smaller the draw number, the closer the runner is to inside the rail and hence a shorter distance to be covered at the turns.

Features we drop:

1. Trainer and Jockey

There are only 24 unique trainers and 63 unique trainers for 14794 races, which makes one-hot encoding impossible and hence we decided to drop it.

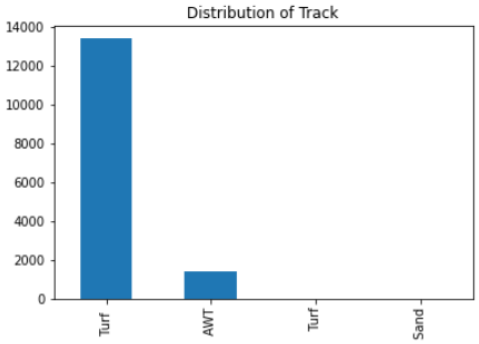


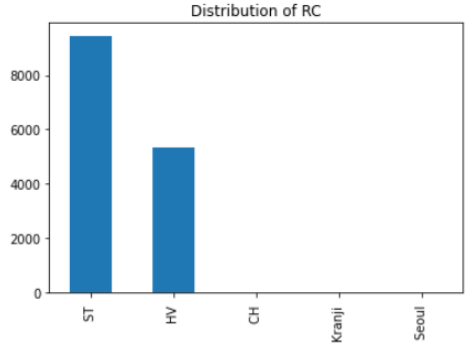




We want to investigate the distribution of some categorical data likes 'Track', 'G' and 'RC' and see if we can handle them using one-hot encoding. The chart below indicates that these columns have an acceptable number of categories and we can apply one-hot encoding after removing some special entry, like RC equals {‘Kranji’ or ‘Seoul’} because of the small sample size.

```
Turf     13380
AWT       1411
Turf         2
Sand         1
Name: Track, dtype: int64
```

```
ST          9451
HV          5328
CH            12
Kranji         2
Seoul          1
Name: RC, dtype: int64
```




### b. Horse_family.csv

Each row of the csv file represents the background of a horse.

Features we keep:

1. Season Stakes
2. Rtg.: rating of the horse at the date of the race
3. One-hot encoding for {Country of Origin, Age, Colour, Sex}

Features we drop:

1. Irrelevant features: {Import Type, Owner, Current Stable Location}
2. Trainer: for the reason mentioned above.
3. Current Rating: overlap with the ‘Rtg.’ above.
4. Start of Season Rating: We only have the rating of the 2020 season, we do not have this information for race takes place in earlier year.
5. Family information {Sire, Dam, Dam’Sire, Same Sire}

The number of unique dam (mother of the horse) and Sire (father of the horse) is large, which makes one-hot encoding impossible.  Hence, we decide to drop these

columns.

```
------------ Sire ------------
count              1293
unique              346
top       Smart Missile
freq                 30
Name: Sire, dtype: object
------------ Dam ------------
count                1293
unique               1261
top       Extension Of Time
freq                    3
Name: Dam, dtype: object
```

```
------------ Dam's Sire ------------
count                1293
unique                454
top       Encosta de Lago
freq                   43
Name: Dam's Sire, dtype: object
------------ Same Sire ------------
count     1293
unique    1144
top        Nil
freq       150
Name: Same Sire, dtype: object
```

### c. Place_dividend.csv

This file contains the win odds for place, since the data are only available after the race, it should not be used to predict the result or added in the model. It should only be used to calculate the reward of the model.

### d. Missing Data handling

Record without the Finish Time implies that the horse did not finish the race because of an accident. Hence the race will not have the ‘place’ and we decide to remove all records with missing value in ‘finish time’.

Some records do not have information on ‘rating’ because they do not have records that affect their rating. We impute them with a starting rating according to their birth origin, suggested by HKJC website.

We then merge the ‘horse_info.csv’, ‘horse_family.csv’ and ‘place_dividend.csv’ which contains the win odds for place into ‘full_data_update.csv’ as the data set for our projects.

## Random Forest Model

We have divided our dataset into a training set and testing set. We have set two flags in our dataset, namely ‘top1’ and ‘top3’ that refers to the place of the race. y1_test and y3_test refer to the flag of ‘top1’ and ‘top3’ in the testing dataset. y1_pred and y3_pred refer to the prediction of ‘top1’ and ‘top3’. We will evaluate our model based on (1) accuracy and (2) net rewards.

Net reward is calculated by the following. The table below is an example for ‘top3’ flag.

If y_pred = 1, we will pay $10 for betting on the horse for winning. If the horse won (y_test = 1), we will get the $(win Odds *10), so our net rewards is $(win Odds –1)*10; If the horse did not win, we lose $10 so the net reward is –10.

If y_pred =0, that means our model predicts the horse will not win, we will not bet on it so the net reward is 0. The calculation is the same for ‘top1’, except that they need to multiple their corresponding win odds when calculating.

| | position | y3_pred | y3_test | top 3 Win Odds | top 3 net |
|---|---|---|---|---|---|
| **9317** | 1 | 1 | 1 | 12.5 | 115.0 |
| **6792** | 9 | 0 | 0 | 0.0 | 0.0 |
| **12640** | 3 | 0 | 1 | 16.0 | 0.0 |
| **13096** | 4 | 1 | 0 | 0.0 | -10.0 |
| **2911** | 3 | 0 | 1 | 19.5 | 0.0 |

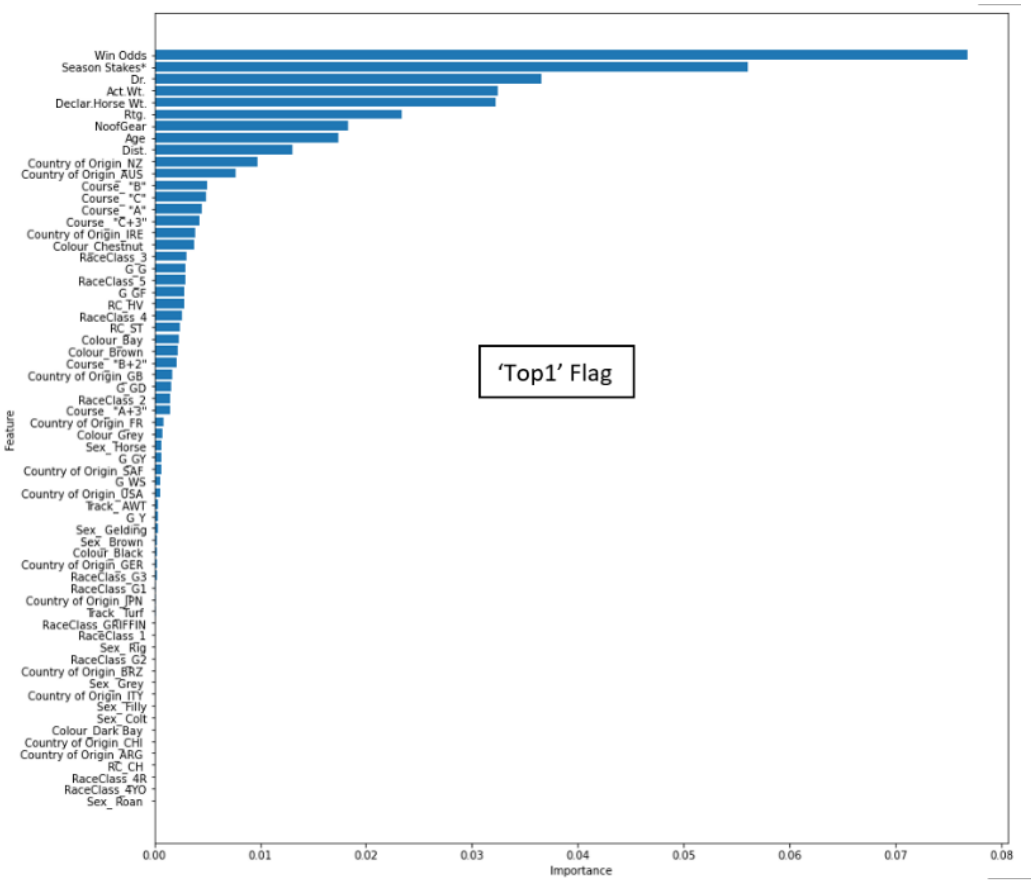
Example of calculating ‘net’

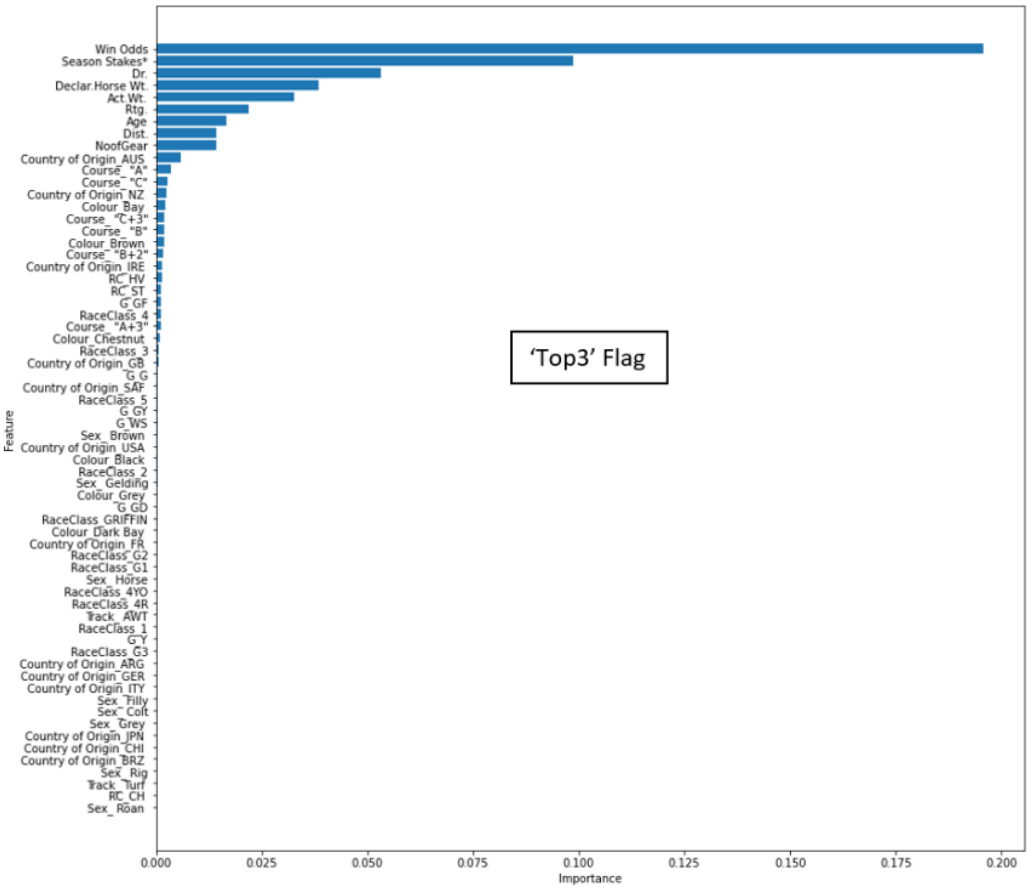
We have built four Random Forest Models on flag ‘top1’ and ‘top3’, we will discuss their performance in the comparison section. They are:

1. Naïve model without features selection.
2. Consider variable importance, features selected by benchmark.
3. Consider variable importance, consider the combination and interaction effect.
4. Consider variable importance, consider the combination and interaction effect, with hyperparameter tuning.

We find out the variable importance using the naïve random forest as the base model. Notice that the minimum importance value is 0. No negative value is found meaning that no columns are found to be noise for our model.

In model (2), we select all the features with importance > 0.01.


Win Odds
Season Stakes*
Dr.
Act.Wt.
Declar.Horse Wt.
Rtg.
NoofGear
Age
Dist.
Country of Origin_NZ
Country of Origin_AUS
Course_"B"
Course_"C"
Course_"A"
Course_"C+3"
Country of Origin_IRE
Colour_Chestnut
RaceClass_3
G_G
RaceClass_5
G_GF
RC_HV
RaceClass_4
RC_ST
Colour_Bay
Colour_Brown
Course_"B+2"
Country of Origin_GB
G_GD
RaceClass_2
Course_"A+3"
Country of Origin_FR
Colour_Grey
Sex_Horse
G_GY
Country of Origin_SAF
G_WS
Country of Origin_USA
Track_AWT
G_Y
Sex_Gelding
Sex_Brown
Colour_Black
Country of Origin_GER
RaceClass_G3
RaceClass_G1
Country of Origin_JPN
Track_Turf
RaceClass_GRIFFIN
RaceClass_1
Sex_Rig
RaceClass_G2
Country of Origin_BRZ
Sex_Grey
Country of Origin_ITY
Sex_Filly
Sex_Colt
Colour_Dark Bay
Country of Origin_CHI
Country of Origin_ARG
RC_CH
RaceClass_4R
RaceClass_4YO
Sex_Roan
Feature
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Importance
'Top1' Flag


Win Odds
Season Stakes*
Dr.
Declar.Horse Wt.
Act.Wt.
Rtg.
Age
Dist.
NoofGear
Country of Origin_AUS
Course_"A"
Course_"C"
Country of Origin_NZ
Colour_Bay
Course_"C+3"
Course_"B"
Colour_Brown
Course_"B+2"
Country of Origin_IRE
RC_HV
RC_ST
G_GF
RaceClass_4
Course_"A+3"
Colour_Chestnut
RaceClass_3
Country of Origin_GB
G_G
Country of Origin_SAF
RaceClass_5
G_GY
G_WS
Sex_Brown
Country of Origin_USA
Colour_Black
RaceClass_2
Sex_Gelding
Colour_Grey
G_GD
RaceClass_GRIFFIN
Colour_Dark Bay
Country of Origin_FR
RaceClass_G2
RaceClass_G1
Sex_Horse
RaceClass_4YO
RaceClass_4R
Track_AWT
RaceClass_1
G_Y
RaceClass_G3
Country of Origin_ARG
Country of Origin_GER
Country of Origin_ITY
Sex_Filly
Sex_Colt
Sex_Grey
Country of Origin_JPN
Country of Origin_CHI
Country of Origin_BRZ
Sex_Rig
Track_Turf
RC_CH
Sex_Roan
Feature
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Importance
'Top3' Flag

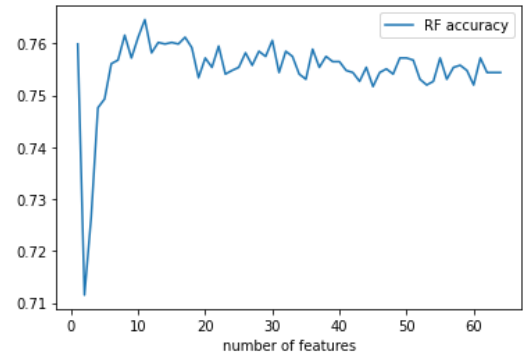
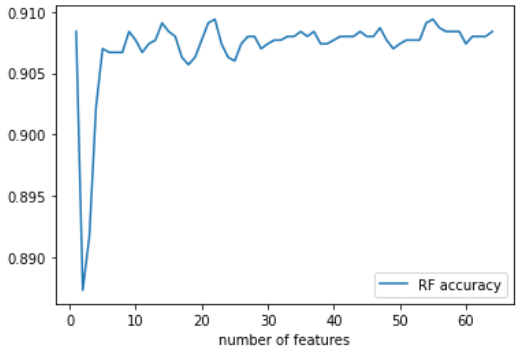
The results above are interpretable and consistent with our beliefs made in the data exploration section.

- Win odds reflect people's expectation on the performance of the horse and it is a good indicator from the general public.
- Season Stakes refer to the winning record of the horse and we can use it to predict the performance of the horse.
- Draw: horses with smaller draw numbers will be advantageous because they are closer to the inner circle and hence they need to run a shorter distance.
- Declare weight: the weight a horse needs to carry will significantly affect its performance.

We consider if the features will have interaction effects, or they are important because of a common feature behind. Hence, we carry out another experiment to implement model 3.

We order the columns by their importance values obtained above, so the first element is the most important, while the last one is the least important.

We first fit the model with the most important features and record their accuracy on the testing dataset. Then we introduce the second-most important features into the model and record their accuracy and repeat the process until all features has been included. The plots below are the accuracy and the number of features included.

| top3 | top1 |
| --- | --- |
|  |  |
| Maximum accuracy = 0.7646 ,it is attained when 11 features are selected | Maximum accuracy = 0.9091, it is attained when 14 features are selected |

In model (3), we include the designated number of features when the maximum accuracy is attained.

For model (4), the features are the same as model (3), but we conduct hyperparameter tuning. The table below shows the changes of the parameters.

| Before tuning | After grid search |
|---|---|
| 'bootstrap': True,<br>'ccp_alpha': 0.0,<br>'class_weight': None,<br>'criterion': 'gini',<br>'max_depth': None,<br>'max_features': 'auto',<br>'max_leaf_nodes': None,<br>'max_samples': None,<br>'min_impurity_decrease': 0.0,<br>'min_impurity_split': None,<br>'min_samples_leaf': 1,<br>'min_samples_split': 2,<br>'min_weight_fraction_leaf': 0.0,<br>'n_estimators': 500,<br>'n_jobs': None,<br>'oob_score': True,<br>'random_state': 0,<br>'verbose': 0,<br>'warm_start': False | 'bootstrap': True,<br>'ccp_alpha': 0.0,<br>'class_weight': None,<br>'criterion': 'gini',<br>'max_depth': None,<br>'max_features': 0.25,<br>'max_leaf_nodes': None,<br>'max_samples': None,<br>'min_impurity_decrease': 0.0,<br>'min_impurity_split': None,<br>'min_samples_leaf': 1,<br>'min_samples_split': 6,<br>'min_weight_fraction_leaf': 0.0,<br>'n_estimators': 250,<br>'n_jobs': None,<br>'oob_score': False,<br>'random_state': 1,<br>'verbose': 0,<br>'warm_start': False |

After hyperparameter tuning, the min_samples_split increased from 2 to 6, meaning that the new model requires a greater number of samples to split the node, which can prevent the issue of over-fiting.

The number of trees decreased after tuning. As Boehmke, B., & Greenwell, B. (2020) suggested in ‘Hands-on machine learning with R’, the search of number of trees starts with 10 times the number of features, in our case that is 11 or 14, which means the number of trees is about hundred and something so the decrease in number of trees is reasonable.

Model Performance and Comparison

| Target: ‘top1’ flag | Number of features | Accuracy | Net |
|---|---|---|---|
| Naive model on ‘top1’ | 64 | 0.9057 | 2 |
| Select features with importance >0.01 | 9 | 0.9026 | 6 |
| Combination: Select the combination of features gives highest accuracy | 14 | 0.9091 | -32 |
| Combination and hyperparameter tuning by grid search | 14 | 0.9029 | 8 |

| Target: ‘top3’ flag | Number of features | Accuracy | Net |
|---|---|---|---|
| Naive model | 64 | 0.7543 | 34933 |
| Select features with importance >0.01 | 9 | 0.748 | 38743 |
| Combination: Select the combination of features gives highest accuracy | 11 | 0.7646 | 41084 |
| Combination and hyperparameter tuning by grid search | 11 | 0.7629 | 36454 |

Interpretation and Discussion:

There is a different trend of accuracy and net reward. The accuracy for flag ‘top1’ is higher than flag ‘top3’, but the net reward for flag ‘top1’ is far lower than flag ’top3’. Accuracy refers to the percentage of number of correct predictions made. For flag ‘top1’ dataset, since only small proportion of the data are classified as ‘top1’ (about 1/14 of them) and majority are not ‘top1’, the variance is small when comparing with flag ‘top3’ dataset, which has larger proportion of data are classified as ‘top3’ (about 3/14 of them). Moreover, it is common that the win odds for ‘place’ for the second and third place is higher than ‘win’, because it is more difficult to predict the first runner up or second runner up when comparing who will be the champion of the race.

Note that model (3) for flag ‘top3’ attains the maximum net reward. One of the possible reasons why the hyperparameter did not help improve the reward is that model 4 reduces the number of trees which the error rate might increase.

Below are the statistics of the testing of model 3:

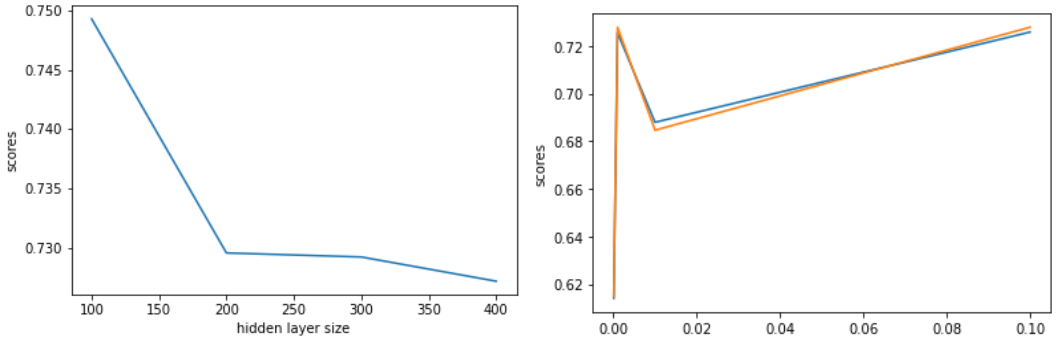
| | |
|---|---|
| Total number of betting chance | 2936 |
| Y3_pred = 1 | 522 |
| Y3_test = 1 | 805 |
| Correct prediction for winning race | 328 |
| Correct prediction for losing race | 1894 |
| Net reward | 41084 |

Out of 805 winning races, our model can predict about 40% of correctly. As for losing races, our model can predict about 88% of that. Overall speaking, we are able to build a model that can gives earning money gambling prediction.

# Neural Network (Approach: Place)

## Experiment & Parameters tuning

The model we are using is the MLPClassifier from sklearn. We first determine the optimal hidden layer size, and then find the optimal learning rate. The results are as follow:




From the above graph, the scores on testing set reached its maximum when the hidden layer size is 100. For the graph of turning learning rate, the scores has a higher value when the learning rate are 0.0001, 0.01, 0.1. Hence, a higher learning rate 0.1 is used.
We then use GridSearchCV to find the optimal parameter, and it is found that the following combination generate the best performance:
solver="sgd", activation="tanh", alpha=0.0001

## Training and Evaluation

After tuning the model, we input the training data to train the model and evaluate its performance using the test set. We assume that the important parameters for the neural network model is similar to those of the random forest model. Hence, we only use the selected features as input. The result is as follow:

| Approach | Train scores | Test scores |
|---|---|---|
| Predicting top 1 | 0.906396388 | 0.901566757 |
| Predicting top 3 | 0.726173239 | 0.72649863 |

From the table above, the performance of the model look pretty good. However, when we checked the prediction result, we found that the model did not learn anything but instead, it guessed 0 for all predictions blindly.
For test set:

| Class | Sum(Actual) | Sum(Predict) |
|---|---|---|
| Others (0) | 2133 | 2936 |
| Top 3 (1) | 516 | 0 |
| Top 1 (2) | 287 | 0 |

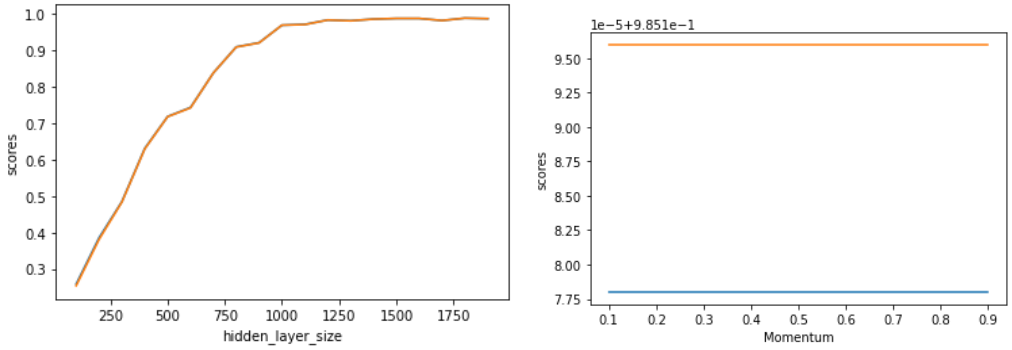
The possible reason causing the model not learning anything will be covered

in the error analysis section. But at this point, the approach of predicting place using neural network seems to be impossible. Therefore, we decided to use another approach: predicting finish time of a horse using a neural network. The following section will cover the result.

# Neural Network (Approach: Time)

## Experiment & Parameters tuning

The model we are using is the MLPRegressor from sklearn. We first determine the optimal hidden layer size, and then find the Momentum. The results are as follow:




From the graph above, the model stops improving since the hidden layer size is 1250. While for the momentum, it seems that it has no effect on the performance of the model, therefore default value 0.9 will be used.
We then use GridSearchCV to find the optimal parameter, and it is found that the following combination generate the best performance:
solver="adam", activation="relu", alpha=0.0001

## Training and Evaluation

After tuning the model, we input the training data to train the model and evaluate its performance using the test set. We use the selected features only just like the place approach model. The result is as follow:

| Train set scores | Test set scores |
|---|---|
| 0.9742523144201924 | 0.9739231543009 |

Again, the model seems to perform very well, but we need to look at the prediction in detail. In order to evaluate the performance on predicting finish time, we calculate the difference between the real value and the predicted value and the result is as follow:

For test set: (unit for time: 1/100 s)

| **count** | **mean** | **std** | **min** |
|---|---|---|---|
| 2937.000000 | 205.099331 | 175.520752 | 0.059530 |
| **25%** | **50%** | **75%** | **max** |
| 76.303223 | 169.350309 | 283.024823 | 1955.495376 |

From the above table, we can see that the model can predict the time close to the real value. The mean difference shows that the model still have 2s error in average but it shows that the model did learn something from the dataset. However, for most races the difference between the winner and the first runner-up is less than a second. Therefore, the performance is still not satisfied in terms of applying on gambling.

# Error Analysis & Limitations

## Unavoidable Bias

An unavoidable Bias is a source of error which is intrinsic to the problem, according to lecture material.

One approach to determine an unavoidable bias, is to compare with human performance on the same problem. Horse racing has a long history and has been well-spread across the world, and it is one of the popular gambling events in Hong Kong. However, it is still a challenging task for a person to predict the winning in a race even with sufficient information provided. Therefore, this problem is considered a complex problem and the optimal error rate may be large.

## Data Distribution

Another reason causing the model not learning from the data is the uneven distribution of class. In the classification approach, we found that most of the data is class 0, which represents that the horse didn't get top 3 position in that race. The counting is as follow:

| | Class 0 (others) | Class 1 (top 3) | Class 2 (top 1) |
|---|---|---|---|
| Counting | 10659 | 2630 | 1388 |
| Percentage | 72.623 % | 17.919% | 9.456% |

From the above table, we can see that the data for class 1 is quite few and class 2 is rare. The model cannot extract patterns regarding class 1 and class 2 and hence it blindly guesses class 0.
One solution is to change the distribution of data. We have tried to do that and found that the scores of the NN model drop to ~0.49, since the solution doesn't change the fact that the data for class 1 and 2 are still insufficient.
To solve this problem or improve the performance, more data is needed. But the HKJC website only provides the horse list in the last three years, which makes it difficult to get more data.

## Misleading Data

In data preprocessing, we merge the table of horse and race together using horse_id. However, we found that one of the important features season Stakes* is not accurate. HKJC website only shows the current season stakes. The history of season stakes of each horse is missing, which means we are using the future data to predict the past result. This may mislead the model significantly.

## Missing Data

In the random forest model, the model performance is evaluated by net gain/loss of betting. The calculation of reward/penalty rely on the win odds provided by HKJC. However, the win odd of some of some race/horse is missing. To deal with it, we fill all the missing values with 0.

If there is no missing data, the net gain of the model should be higher as it will get more reward when it successfully predicts the place.

## Conclusion

We have tried with different types of models. The NN model performance is not satisfied even with a high score. The NN model shows us that the accuracy of the prediction given by the model may not relate to the performance of the model and we need to consider the data distribution.

The Random Forest model is able to generate profit, and its features are interpretable. Yet, there are several issues including unavoidable bias, data distribution, misleading data and data missing. These issues reduce the performance of the model and some of them are hard to fix. Since most of the issues are about the data source, it is believed that a more rigid data collection can help the model perform better.