Report on Horse Betting Strategy

# An initial look into the data

The dataset provided contains information on 3208 bets placed on horse races by Mustard Systems. None of the 8 features nor the target column contain any missing values, however the results column does contain some ‘NON-RUNNER’ entries resulting in the bet being cancelled refunded. These will be removed before creating the model as they provide no useful information on the outcome of race and do not affect the profits, leaving 3179 samples to be inspected.

# Exploratory Data Analysis

Chart

Description automatically generatedFirst the profit of is calculated using the odds and stake of each bet and set as a new column in the dataframe.

Figure 1: The probability frequency function of the profit per bet, in bins of £4.

Plotting a probability frequency graph of the profit per bet, it seems that the profit has a positively skewed distribution. However, as none of the stakes are more than £5 and so the greatest loss is also £5, it will be assumed that this distribution is normal, for simplicities sake. As previously stated, the mean profit is £1.63, and the standard deviation is £12.96, so the confidence interval can be calculated.

It can now be said with 95% confidence that the true mean lies within the interval (1.18, 2.08). As both of these values are positive, it means that the current strategy used by Mustard Systems is a good one as it should be resulting in an overall profit, but there may still be room for improvement.

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generatedWin rate is defined as the percentage of bets placed with a positive profit returned. Whilst a bet placed on a chase race tends to win more often (50.5% of bets), Figure 2 shows that there is not a great difference between hurdle and flat races. One possible explanation for the chase race to have the highest win rate is due to the fact that it is generally considered to be the most technical race out of the three, and so there is less likely to be an ‘upset’ with an unexpected horse winning. This feature could be removed when creating the model as it may generate unnecessary noise during training resulting in underfitting.

Figure 2: A bar chart demonstrating how the race type affects the win rate.

Chart, bar chart

Description automatically generatedWhilst one would assume that the greater the bet strength, the more likely the bet is to be successful, Figure 3 demonstrates that this is not the case. A bet strength of 4, not 5, is actually the most likely to win, at a calculated average of 49.5%.

Figure 3: A bar chart demonstrating how the bet strength rating affects the win rate.

Chart, line chart

Description automatically generatedThere is no one surface condition (going) which provides a substantially better win rate over others, however a heavy graded surface has by far the lowest win rate at 35.8%. This may be because very wet conditions lead to an increase in unpredictability of the winner, and therefore to optimise the betting strategy, Mustard Games should avoid bets on heavy graded surfaces. Bets placed on good, firm, good/firm and fast surfaces should be the focus, with all resulting in a win rate of above 45%.

Figure 4: A bar chart demonstrating how the surface conditions (going) affects the win rate.

Figure 5: A line plot demonstrating how the number of horses running in a race affects the win rate.

Figure 5 shows the effect the number of runners in each race has on the win rate, which has been fitted with a polynomial line of best fit of order 2. From this plot it is clear to see that the value with the highest win rate is 14 runners at a value of 51.8%. Races with 12-15 runners inclusive should be the focus of Mustard Games as they all have win rates of more than 50%.

Chart, scatter chart

Description automatically generated

Several pieces of information can be understood from Figure 6. To start with, the least squares trendline depicts a negative coefficient meaning that in general, the more days since a horse last raced the less likely a bet is to be successful. Whilst the variance in win rate is large for all values, it becomes more pronounced after roughly 65 days since the last race. As the coefficient of the trendline gradient is of a small magnitude, this feature is not too important when considering what bets Mustard Games should place in the future, but a good strategy would be to avoid races where the horse has not run for more than 65 games.

Figure 6: A scatter plot demonstrating how the number of days since a horses’ last race affects the win rate.

Chart, scatter chart

Description automatically generated

Similar to the days since a horses last race, the maximum amount that can be staked before the odds change shows a comparable negative gradient. Figure 7 shows the least squares trend line predicting a win rate of 46.6% at a maximum bet of £2, and all the way to 28.9% at £2150. As a result of this, bets with a smaller maximum bet should be taken as they will be successful more often.

Figure 7: scatter plot demonstrating how the maximum available bet before an odds change affects the win rate.

# Most profitable betting opportunities

Using the criteria found in the above section, we can inspect a small subset of bets that seem to be most profitable:

* 13 or 14 runners
* Chase race type
* Bet strength of >= 2
* Surface grade (going) of good, firm, good/firm or fast

This criterion will be narrow enough to only contain the most profitable bets, but wide enough to provide a large enough sample size that will keep variance down. This gives a 50% confidence that the average mean profit of these bets will be in the interval (0.780, 7.943). As both ends of the interval are positive, it is more likely than not that each bet will result in a profit, and so that maximum stake on bets within this criterion should be staked. More samples are required to give an improved confidence level.

# Creating the model

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Figure 8: Bar charts comparing the accuracy and F1 scores of different classification models for this problem.

First the data is split into training, validation and testing sets in the ratio of 80:10:10 before a logistic regression model is fitted in order to calculate a baseline score to be improved upon. The classification ensembles Random Forest and XGBoost will be used as a comparison to the baseline model and then the best model selected and used for the remainder of the project. After tuning the hyperparameters of each model using GridSearchCV with a KFolds cross validator, the accuracy and macro average F1 score (which takes both precision and recall into account) of each model can be compared. These metrics will be saved in a .json file in the model directory in case of the need for inspection at a later date. As is demonstrated in Figure 8, using the XGBoost Classifier has both a greater accuracy and F1 score than the other two models, and so will be used going forward.

## Testing the model

Whilst accuracy gives information on how often a model predicts the true outcome, it is not always the best metric to use. This is especially prevalent when the dataset is unbalanced, as the class with the highest number of records (losing races in this case) will dominate the calculation. False negatives are of a less importance, as no money is lost here – instead precision and recall will be inspected. Precision is a measure of the probability that an event classified has been correctly classified by the model, or the proportion of true positives to total predicted positives. On the other hand, recall is the proportion of true positives to total real positives, which in this case represents the number of winning races the model successfully predicted. The precision score is probably of a greater importance here, as false positives will result in a stake lost, whereas false negatives are simply the money that could have been won. By inspecting the testing set, the following metrics are calculated.

* Precision: 0.749
* Recall: 0.749

Chart, treemap chart

Description automatically generatedThese values are identical which means the model has predicted the same number of false positives as false negatives. This information can also be portrayed by a confusion matrix in Figure 9.

Figure 9: A confusion matrix for the testing set predicted by the XGBoost Classifier model.

Chart

Description automatically generatedBy plotting the models’ feature importance, features that hold the most weight within the model can clearly be identified. Without much surprise, the odds of the bet is the most important factor, followed by the number of runners and the maximum bet available. Interestingly, the bet strength rating is one of the less important factors, perhaps indicating that the algorithm used to generate this value needs re-tuning.

Figure 10: A bar chart demonstrating the importance of each feature used by the XGBoost Classifier model.

A bet is coming up with the following features:

* £400 available before odds change
* Odds of 10
* 8 runners
* Flat race type
* Firm graded surface
* 20 days since the horse last raced
* Bet strength rating of 4

After entering this information into the model, it predicts the probability of the horse winning the race is 36.5%. The expected return on a bet is calculated by:

With odds of 10, this means that winning this singular bet would result in a 1000% return, and so the expected return off this bet is 365% of the staked amount. As this number is greater than 100%, the bet should be taken as it is likely to result in a profit for the company.

# Evaluation

Given more time, there are several things that could be improved with the report and the model created.

1. A Tableau dashboard to give a concise visual insight into the data provided.
2. Use a bootstrap resampling method to calculate confidence levels rather than assuming a normal distribution.
3. One-hot encoding, rather than label encoding the categoric values in the design matrix.
4. Consider removing certain rows from the matrix to improve the model, such as bets with extremely high/low odds.
5. Consider removing certain features with low importance, such as race type and days since the horses’ last race to combat overfitting.
6. Creating an artificial neural network model using PyTorch and compare to Scikit-learn’s classification models.