Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Mikko Juola

Feasibility of building a profitable data-driven betting system on horse races

Master's Thesis

Espoo, May 7, 2018

Supervisor: Aristedes Gionis, Professor

AALTO UNIVERSITY
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF THE
MASTER´S THESIS

Author:Mikko Juola

Title of the thesis: **Feasibility of building a profitable data-driven betting system on horse races**

Number of pages:62

Date: May 7, 2018

Major:
Computer Science

Supervisor:
Aristedes Gionis

This thesis examines the feasibility of building a profitable betting system that bets on horse races, using historical racing data between 2010 and early 2018. Historical precedent for profitable, data-driven betting systems exist in literature but they mostly focus on pre-2000 data. We combine two datasets sourced from Betwise.co.uk and Betfair to produce a diverse set of features and train a Plackett-Luce-based model to obtain accurate estimations of probabilities that a horse will win in a race. We also perform automatic feature analysis on our engineered features in order to understand what factors are important in the task of predicting horse race results. We find that a model that is designed to be combined with publicly posted betting odds can theoretically achieve meager positive returns. Our results also suggest that empirical testing of the betting system in this thesis would require over 6 months of betting before it could be confidently judged profitable.

Keywords: sports betting, learning to rank, plackett-luce, gambling

Publishing language: English

AALTO-YLIOPISTO
Perustieteiden korkeakoulu
Master's Programme in Computer, Communication and Information Sciences

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Mikko Juola

Työn nimi: **Feasibility of building a profitable data-driven betting system on horse races**

Sivumäärä:62

Päivämäärä: 7. toukokuuta 2018

Pääaine:
Computer Science

Valvoja:
Aristedes Gionis

Tämä diplomityö tutkii, onko tuottoa tekevä raviurheiluun kohdistuva vedonlyöntijärjestelmä käytännössä toteuttamiskelpoinen. Työssä käytetään ravidataa vuosien 2010 ja 2018 väliltä. On olemassa edeltäviä tutkimuksia, joissa tuottoisia vedonlyöntijärjestelmiä on toteutettu mutta niissä käytetty data koskee pääosin aikaa ennen vuosilukua 2000. Tässä työssä yhdistetään dataa kahdesta eri lähteestä: Betwise.co.uk sivustolta ja Betfair -vedonlyöntiyhtiöltä ja tämä data muutetaan laajaksi ja monimuotoiseksi joukoksi syötteitä. Syötteitä käytetään Plackett-Luce pohjaisessa mallissa, joka tuottaa tarkat arviot todennäköisyyksistä siitä, että eri hevoset voittavat ravin. Työssä tehdään myös automaattinen syötteiden arviointi siitä, mitkä syötteet ovat tärkeimpiä ravien tuloksien ennustamisen kannalta. Parhain työssä luotu malli on teoreettisesti niukasti tuottoisa, jos se yhdistetään julkisten vedonlyöntikertoimien kanssa. Tuloksissa havaitaan myös, että empiirinen kokeilu vedonlyöntijärjestelmän kanssa vaatii yli 6 kuukautta vedonlyöntiä, ennen kuin tuloksiin voi luottaa.

Avainsanat: sports betting, learning to rank, plackett-luce, gambling

Kieli: Englanti

Contents

1 Introduction

The main point of this master's thesis is to build a profitable betting system. One of the sub-goals is to understand horse races and answer a number of questions along the way, such as: what factors are important in predicting horse race results? How should we design a system that predicts horse race results? Is a profitable betting system possible at all?

The conclusion that we will find by the end of this thesis is that it is indeed possible to bet profitably when equipped with a suitable data set. However, the edge over the public betting market is very small. Moreover, our estimations over how the betting system would perform in the real world have some caveats on them that can mean they are overly optimistic and we cannot have full confidence in them. We also find that if we wanted to perform an empirical experimentation of our system, it would take months to collect enough data to draw conclusions, if our system is really profitable.

There have been several efforts in the past to build profitable betting systems based on historical data and many claim to be successful. For example, Hausch et al., [15] back in 1981 reported positive returns for a betting system that relied purely on currently posted market betting odds. In 1998, Mukhtar, M. Ali [2] examined some prominent data-driven betting systems at the time and found out that none of them were profitable over long-term. He also suggested that the results obtained by Hausch et al. may have been spurious or a peculiarity of the data set they were working with; a suggestion we find plausible, given our own tests on more recent data set.

There are more examples of research that claim successful betting systems. One of them is Benter's [4] system based on training a type of logistic regression model using a diverse set of features. Benter used Hong Kong horse races between 1986 and 1993 as his input data. His betting model achieved better goodness-of-fit in terms of predicting of horse races results than betting public and significantly better when he combined public betting odds and his model together. For this thesis, we are using a very similar loss function as Benter, based on Plackett-Luce ranking but we did not get results that are even close as good as Benter's. It could be that between 1986 and 1993 the betting public was not very sophisticated in their estimations and building a profitable betting system based on data was much easier. It could also be that betting profitably in Hong Kong is easier than betting in Ireland and United Kingdom. We use some of Benter's ideas in our feature engineering.

This master's thesis is yet another effort to build a horse race betting system but this time using data between 2008 and 2018. Ultimately, even with extensive feature engineering, our system only gains a very slight edge over the market.

There are two main themes contained within this thesis. The first theme is about understanding horse races. Chapters from 3 to 4 are about analyzing different features of horse racing and validating some "common wisdom" factors about horse racing. The second theme is about building a betting system, that can, at least in theory, bet and be profitable. This system is developed and discussed in chapter 5 and 6.

We make a number of contributions.

- Certain "common wisdom" factors of horse racing are analyzed for usefulness in predicting results, in particular, draw bias and pedigree. We find that draw bias and pedigree, while useful in certain situations, are not very useful predictors of horse race results. Other, easily computable factors are much more predictive.
- We examine how smart the market betting odds are before a horse race begins. We find that there is a surge of information coming in during last 5 minutes before the race begins.
- We discuss several loss functions and metrics that can judge how effective some predictor is at ranking horse race results. We find that Plackett-Luce loss function has several desirable properties. Plackett-Luce is particularly useful in overcoming overfitting when sample sizes are small.
- We analyze a diverse set of features for their usefulness in predicting horse race results. We find that most useful features tend to be ones that take past performance of a horse, jockey or trainer into account.
- We build a stacking model that improves upon betting market odds by stacking our predictions on top of them. We backtest a betting system that, in theory, should be profitable. The edge over the market is small.

The betting system and all experiments are, for the most part, computed with a codebase written in Haskell, with data stored in PostgreSQL. Various models are tested and trained using gradient descent. All gradient-based optimization tasks are done using automatic differentiation, in particular the *ad*[1] automatic differentiation library. We will not be focusing on software implementation in this thesis but we will refer to it occasionally. We found that Haskell and its automatic differentiation is particularly flexible when it comes to quickly prototyping new experiments. This flexibility comes at the cost of performance and our software choice has cost us the ability to run some experiments, in particular we could not run an extensive out-of-sample testing for our betting strategy (we will discuss this in section 6.1.3). Most of the charts in this thesis are made with another Haskell library, *Chart*[2].

2 Background and prior research

There are several academic fields that have developed relevant theory for betting systems. Financial economics is relevant because the betting markets are in many ways similar to stock markets and much of the same academic framework can be applied. Machine learning is relevant because many betting systems rely on making predictions using a diverse set of features. One branch of machine

[1] https://hackage.haskell.org/package/ad
[2] https://hackage.haskell.org/package/Chart

learning is particularly relevant: ranking problems. In a horse race, horses finish in a specific order. Ranking models can be useful to model this behavior and make use of the ranking information in making new predictions. Gambling also has connections to information theory.

2.1 Research on betting systems

Many of the assumptions about the behavior of betting markets relies on the *efficient market hypothesis.* This concept was brought forth in a seminal paper by Fama in 1970 [13]. The efficient market hypothesis (EMH) states that the price of an asset reflects all available information thus far; making it impossible to "beat the market". There are several levels of efficient market hypothesis. For example, we can distinguish between EMH that assumes the price reflects all available *public* information versus EMH that assumes the price also reflects public and *private* information. EMH is usually discussed in the context of financial markets, e.g. the stock market.

The efficient market hypothesis is commonly believed to apply for horse race betting as well. The difference between the stock market and a bet is that the value of a bet is unambiguously decided when the horse race is over. If EMH applies to horse race betting, it means the public betting odds reflect all available information so far; the odds on each horse is a sophisticated estimation that the horse will win the race.

Ali formally investigated the efficient market hypothesis on horse race betting in 1979 [1] and empirically confirmed that the betting odds accurately reflect actual probabilities of winning.

Since 1980s, several researchers have attempted to build profitable betting systems. Bolton and Chapman in 1986 [6] built a logistic regression model to predict horse race results. They assumed that the performance of each horse in a horse race is determined by the sum two terms: a deterministic performance value and an error value that is independently distributed from other horses in a race. The loss function they used for optimizing appears to be Plackett-Luce although they do not call it by that name. Their work claims positive returns but their sample size is only 200 races. In comparison, our data set in this thesis contains approximately 110000 horse races.

The same author, Chapman, released a second paper in 2008 [9], with improvements over their 1986 research. They increased the number of features used for prediction and used a sample size of 2000 races. The conclusions in this study is that it is possible to gain substantial results from betting on horse races, using data-driven models, for Hong Kong races.

Not all prior research claims successful betting returns. In 1980, a relatively well-cited paper by Hausch et al. claimed to have invented a profitable betting model [15]. However, in 1998, Mukhtar, M. Ali [2] examined this result, along with some other prominent data-driven betting systems at the time and found out that none of them were profitable over long-term. He suggested that the results obtained by Hausch et al. may have been spurious or a peculiarity of the data set they were working with.

Another research on Hong Kong races is by Benter [4]. Benter finds that a model that combines the public betting odds on the market with predictions generated by a logit model can yield substantial returns.

The main platform for betting that we use in this thesis is the Betfair betting company. Several research papers investigating market efficiency and information use Betfair data in their analysis. Croxson, in 2014, [12] examined how quickly market adjusts after a goal is scored. Croxson's conclusion is that market responds rapidly, extremely quickly after a goal is scored and that it is not possible to exploit any systematic drift in the system. Given that this result was obtained on the same platform as our work (Betfair), we likely can surmise that very quick response to new information is true for horse racing as well.

Gamblers with access to private information may be able to gain an edge over the public.. Brown [8] examined horse racing trading odds on Betfair and discovered that some portion of the gamblers seem to have access to information before the betting public. Betfair gives information about the extent of trading volume and this can be used to gauge how many bets are being placed at any given time. Brown noticed that some amount of bets are placed extremely quickly after an event, with more bets trickling down over 5 minutes; an observation that is consistent with his hypothesis.

We find that there are not many research papers that report unsuccessful attempts at building a profitable betting system. We also note that it is difficult to find attempts at profitable betting models after 2010. The past systems may have been profitable because the betting public may not have been quite as sophisticated as it is today. The model we use in this thesis is very similar with Benter's model and Chapman's model; we even use the same loss function (Plackett-Luce) but we did not find it easy to beat market odds.

2.2 Betting and information theory

Information theory and gambling are linked. In particular, one of the most useful insights in gambling is by J.L. Kelly [16] who formulated the *Kelly criterion.* Kelly criterion can used to choose the optimal bet size. If b is the betting odds, being wagered and $P(\text{win})$ is estimated probability of win in an event, Kelly criterion yields that optimal bet size if the fraction f of current bank.

$$f = \frac{bP(\text{win}) - P(\text{lost})}{b} \tag{1}$$

$$P(\text{lost}) = 1 - P(\text{win})$$

Kelly formulated his idea in terms of information technology. The key idea is that that the goal is to maximize the expectation of the logarithm of the capital over time, rather than just expected profit from any single bet.

Kelly's strategy generalizes to choosing a distribution of bets over many horses in a horse race. A good introduction to this is in the book Elements of Information Theory, by Cover and Thomas [11] where they discuss the following concepts and strategy, based on Kelly's work.

Let $S(X)$ be wealth after a horse race X, after all bets are placed, relative to wealth before the race (e.g. a profitable betting strategy might have expected wealth $\mathrm{E}(S(X)) = 1.01$, i.e. your expected wealth grows by 1% from starting point after each race). Then, your *doubling rate* is:

$$W(b, p) = \mathrm{E}(\log\ S(X)) = \sum_{i=1}^{n} p_i \log(b_i o_i) \tag{2}$$

Where b is a vector of bet allocations ($\sum_i b_i = 1$, $b_i \geq 0$), p is a probability vector that contains a probability of each horse that they are going to win and o is a vector of rewards; if horse i wins, wealth increases by o_i.

If horse race results are assumed to be independent (i.e. the results of a horse race happening tomorrow do not depend on results of a horse race today), then *proportional betting* is the optimal strategy. This means $b = p$. In other words, bets should be allocated exactly according to the winning probabilities. This also implies you never risk your entire wealth, because you are going to place some of your wager on every horse (assuming your estimation of p is does not contain mistakes).

Cover and Thomas also show that difference in doubling rate equals the mutual amount of information between horse race X and new information Y. Let $W^*(X)$ be optimum doubling rate (i.e. W from equation 2 where we set $b = p$). Then:

$$W^*(X|Y) - W^*(X) = I(X; Y)$$

Where I is the measure of mutual information between in bits.

In section 6.1.3 near the end of this thesis, we build a type of betting strategy based on Kelly criterion.

3 Methodology

In this chapter, we establish our measurement metrics, our data sources and our benchmark metrics. First, we will review some ways to judge how "good" the predictions of some horse race prediction system are, until settling on Plackett-Luce loss and goodness-of-fit. After we have discussed measurement, we will introduce the data sources used in this thesis. We will then discuss a concept called *starting prices,* a type of betting odds as they are when a horse race begins. The starting prices are a good benchmark to how strong other predictors are; we will use our metrics to compute how well they perform.

3.1 Optimizing towards best rankings

Let us say we have created a new predictive model for horse races. Our system outputs a probability for each horse in some race that they are going to win the race. For example, let us say our probabilities for horses H_1, H_2, H_3 are $\{0.2, 0.4, 0.2\}$. First trouble with our predictor would be that the probabilities

do not sum up to one: $0.2 + 0.4 + 0.2 = 0.8$. But that shouldn't be a difficult problem to fix; we can normalize it: $\hat{P}_k = \frac{P_k}{\sum_{i=1}^{n} P_i}$ and now the new probabilities are: $\{0.25, 0.5, 0.25\}$. The output is a proper *probability vector*. In this thesis, we normalize all predictions this way unless way specify otherwise.

Continuing with the example above, let us say that horses H_1, H_2, H_3 finished first, third and second, respectively. We can denote this with function ϕ that outputs the actual finish position of a horse: $\phi(H_1) = 1, \phi(H_2) = 3, \phi(H_3) = 2$. In other words, ϕ defines a *permutation* of the horses, where the permutation tells us in which order the horses finished the race.

The question that this chapter is concerned with is how will you judge how well your estimates for each horse race are holding up. The example above predicted {0.25,0.5,0.25} for our hypothetical race but the horse, for which we predicted 50% probability of winning, highest of the three horses, actually came in last. We want to find a measurable, comparable way to put a number on the "goodness" of this prediction. Another goal is to find a good metric to optimize for, using gradient descent.

3.2 Winner-predicted-correctly metric

The simplest, most obvious way to measure this is to pick out the prediction for which we predicted the highest score and see if they actually won the race. Our "score" is 1 if the horse won, 0 otherwise. We can take the average of our predictions to see what is the percentage of races where we predicted the winner correctly.

While this works, it misses out on a lot of information. Let us say our horse race has 15 horses racing in it and you predict horse H_{12} is the most likely winner of this race. It turns out H_{12} came in second. The score is 0. However, you would also be given zero score if H_{12} came in at last place. Finishing second out of 15 runners is much better than finishing last; yet this simple scoring function cannot take that into account. In other words, we needlessly "punish" models that get it mostly right and consider them just as bad as models that get it mostly wrong entirely. To top it off, for optimization tasks this throws away a lot of information about each race (the ordering of horses is not considered at all) and in effect is like working with a smaller data set.

Moreover, good predictors often agree about who is most likely going to win a race. The part they disagree with is the *probability* that the horse is going to win the race. If you are comparing predictors for horse racing this can hide information about subtle differences between different predictors and make good predictors look extremely similar.

Because of all these drawbacks, we find this measurement is not an appropriate metric to judge our predictions. Table 1 that lists winner-predicted-correctly percentage of BSP, ISP and Forecast prices. BSP stands for Betfair Starting Price, ISP stands for Industry Starting Price and Forecast price is a racecard price several days before the race begins; all are sophisticated estimations of race horses winning a race (we will discuss what these prices are in more detail in section 3.7 on starting prices).

Price	% Winner predicted correctly
BSP	33.7%
ISP	33.5%
Forecast price	29.4%

Table 1: Winner percentage of favorites as determined by BSP, ISP and Forecast prices.

3.3 Root-Mean-Squared-Error

Another popular measurement used in regression and optimization tasks is the root-mean-squared-error, abbreviated as RMSE. This can be a marginal improvement over binary winner-or-not measurement.

For RMSE, we need to cast the horse racing prediction problem as a binary classification problem. This is easy; just consider winners to have label 1 and losers to have a label of 0. The prediction system outputs a score for every horse in every race they participate in and we compute RMSE against the labels. Mathematically, with P as our prediction system that outputs a score between 0 and 1 and W as an indicator function that equals 1 if the horse given as argument won and 0 if it lost:

$$W(H_i) = \begin{cases} 0 & \text{if } H_i \text{ lost the race} \\ 1 & \text{if } H_i \text{ won the race} \end{cases}$$

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(P(H_i) - W(H_i))^2}$$

This is somewhat better than just binary judgment; the RMSE will not punish bad predictions quite as harshly. However, this still throws away information about ranking since we only consider all observations to be winners or losers with no middle-ground. RMSE will still harshly punish predictions that are almost correct but not quite (e.g. if we predicted that a horse will win with 0.75 probability but came in at second place in a 15-horse race; that is still a relatively good prediction, but it is judged the same way as if that horse came in last). Additionally, something we found out during measurement, is that RMSE scores for different predictors tend to be very close to each other and this makes it hard to meaningfully distinguish them; in this regard it appears to be even worse than just considering the number of cases where the winner was predicted correctly. In order to drive the point: in table 2 we show RMSE for BSP, ISP and the Forecast prices. The ISP and BSP prices are extremely close to each other. We think this is because of two factors: 1) There is a large number of losers in the data set compared to winners; correct answer is 0 for most runners and most models get this right easily 2) Horse races are fundamentally highly unpredictable, it is rare for horses to actually have a high chance of winning a race. RMSE scores can never get very high in this setting.

Price	RMSE
BSP	0.2874
ISP	0.2877
Forecast price	0.2947

Table 2: RMSE for BSP, ISP and Forecast prices

Most alarmingly, Forecast prices get a higher RMSE score than BSP or ISP. The forecast prices are much less sophisticated estimations of the horse race results than either BSP or ISP, so this really shows that RMSE is not a good choice for judging horse race ranking models.

3.4 Kendall tau distance

Predictions systems for horse racing usually can produce a ranking of the horses, in the order they are expected to finish the race. We can judge the quality of this ranking by comparing how "far" away the predicted permutation is from the actual permutation that the horses finished the race in. One way to compute this is something called *Kendall tau distance,* developed by Maurice Kendall [17]. Letting ϕ_1 and ϕ_2 be permutations (or equivalently, lists), the mathematical formulation of Kendall tau distance is below:

$$K(\phi_1, \phi_2) = |\{(i,j) : i < j, (\phi_1(i) < \phi_1(j) \wedge \phi_2(i) > \phi_2(j)) \vee (\phi_1(i) > \phi_2(j) < \phi_2(j))\}|$$

The intuition of Kendall tau is easier to explain than its mathematical formula. Kendall tau yields 0 if the permutations are equal. It yields $\frac{n(n-1)}{2}$ if the permutations are the exact reverse of each other. It is also equal to the number of swaps you would have to perform on one of the permutations to bubble-sort it to match the other permutation. You can normalize Kendall tau to have a value between 0 and 1 by dividing it with $\frac{n(n-1)}{2}$ and subtracting it from 1; so that 0 means completely reversed ordering and 1 means equal ordering.

Kendall tau is somewhat more insightful than just considering the winner but it is non-differentiable and thus more challenging to use in optimization tasks, that often rely on gradient descent. Still, it can be used as one metric to compare different horse race predictors. The main advantage of Kendall tau is that it is intuitively one of the easiest to explain and understand.

In figure 1 there is a type of graph computed using Kendall tau with four predictors: the BSP prices, the ISP prices, Forecast prices and a model that randomly assigns a permutation. First, we compute Kendall tau for every horse race, comparing the predicted permutation to the actual finished permutation. Then we sort all the races according to their Kendall tau number. Then we plot a line, where y-axis is at the Kendall tau number and x-axis goes through each race in the order of Kendall tau numbers. This graph tells the portion of how many races have Kendall tau numbers of certain magnitude.

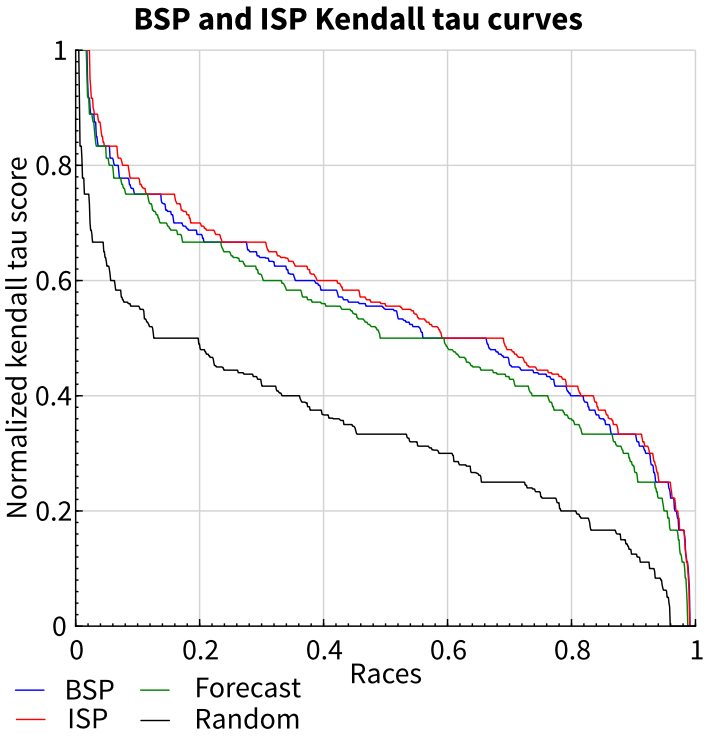

Figure 1: Kendall tau for BSP, ISP and Forecast prices. The lines are non-smooth because there is a finite number of possible Kendall tau values for our races in our data set.

Better predictors should have larger number of high-scoring Kendall tau races. This means there are more races that are predicted correctly. And indeed, it is clear that BSP and ISP prices lead the pack over Forecast and a random predictor.

The only real issue for our purposes with Kendall tau is its non-differentiability. And for that reason, we also set Kendall tau aside in favor of Plackett-Luce, which we will discuss next.

3.5 Plackett-Luce

We now come to the first measurement of ranking that has a more probabilistic viewpoint. This model is the Plackett-Luce ranking, which we will refer to as P-L model or loss, depending on context. This loss function is the result combining the work by two different authors: Luce in 1959 [20] and Plackett in 1975 [22]. In particular, the Plackett-Luce makes an independence assumption that reflects Luce's choice axiom: in a pool of N items, the probability of choosing item I_k over I_j is not affected by the addition or removal of other items in the pool.

To start off, the mathematical formula of Plackett-Luce loss function, when applied to horse racing, is as follows:

$$\mathrm{PL}(H) = \prod_{i=1}^{n} \frac{S(H_i)}{\sum_{k=i}^{n} S(H_i)} \tag{3}$$

Where $S(H_i)$ is the estimated probability that horse H_i is going to win the race. The subscripts refer to the order the horses actually won the race (so H_1 refers to the winner of the race and H_n refers to horse that came in last). In equation 3, $S(H_i)$ must be a positive real number. A slightly altered formulation can relax this requirement by doing exponentiation on all $S(H_i)$ values and then the estimations can be given as any real number; P-L will "normalize" them. This altered formulation is below:

$$\mathrm{PL}(H) = \prod_{i=1}^{n} \frac{e^{S(H_i)}}{\sum_{k=i}^{n} e^{S(H_i)}} \tag{4}$$

$S(H_i)$ can be substituted with some model that outputs any real number. One choice is linear regression. For example, if we have a hypothetical linear regression model with weights $w_{0,}w_1, w_2$ and a and b as input values to linear regression (that are different for each horse), Plackett-Luce can be formulated like this:

$$\mathrm{PL}(H) = \prod_{i=1}^{n} \frac{e^{w_0+aw_1+bw_2}}{\sum_{k=i}^{n} e^{w_0+aw_1+bw_2}}$$

We use linear regression and simple feed-forward neural networks in this thesis in place of $S(H_i)$, depending on the task at hand.

The intuition behind P-L is also easy to understand. It computes the probability that a certain permutation is seen, given probabilities for each horse that they will win the race. For example, using the example given at the start of this section, we have a horse race, with three horses, and we predict probabilities $S = \{0.25, 0.5, 0.25\}$ for these horses, and they finished at 1st, 3rd and 2nd places respectively. P-L loss in this case is (using equation 3, with ϕ being the actual finished order):

$$\mathrm{PL}(H) = P(S|\phi) = \left(\frac{0.25}{0.25+0.5+0.25}\right)\left(\frac{0.25}{0.5+0.25}\right)\left(\frac{0.5}{0.5}\right) = \frac{1}{12}$$

This means that getting the permutation $\phi = \{1, 3, 2\}$ with winning probabilities $\{0.25, 0.5, 0.25\}$ has a 1/12 chance of happening. P-L makes some assumptions about the process; the primary assumption is that the underlying events are independent of each other (i.e. the Luce axiom). Casting this axiom to horse racing: there is an assumption that if we remove any horse from the race it will not affect the performance of any of the remaining horses, relative to each other. P-L removes a horse from the equation and then considers the

remaining horses by normalizing their probabilities back to 1, as if the first horse was not even present. If, however, the performance of horses in a race are not independent (as likely can be expected), P-L will end up being a little off.

This brings the main problem of Plackett-Luce loss. It is typical that a horse race has some hopeless runners that have a very low probability of winning. Let us say we have 15 runners in a race and the three last horses have probabilities of 0.003, 0.002 and 0.001 of winning. At the end of P-L evaluation, this would end up being normalized to $\{0.5, 0.33, 0.17\}$ in the last terms of P-L. The noise in tiny probabilities can be expected to be much higher than in higher probabilities and by normalizing the tiny probabilities to sum up to 1, at the end of Plackett-Luce computation, we greatly enhance this noise.

To mitigate this effect, P-L can be truncated. This is also the approach taken by Benter [4] in his betting model. Instead of computing P-L all the way to the end, we will only consider the first N terms. In mathematical terms, we can define PLT (based on equation 3):

$$\mathrm{PLT}(H, N) = \prod_{i=1}^{\min(n,N)} = \frac{S(H_i)}{\sum_{k=i}^{n} S(H_i)} \tag{5}$$

The only difference between equation 5 and 3 is that the number of multiplications performed in the product is capped at N. There is no obvious way to choose N without experimenting; in this thesis whenever we optimize using P-L the optimal N is found by cross-validating. In this thesis we refer to N as the "depth" of P-L. For example, "P-L to the depth of 1" means we only compute the first term in P-L, i.e. $\mathrm{PLT}(H, 1) = \frac{S(H_1)}{\sum_{k=1}^{n} S(H_i)}$.

3.5.1 Plackett-Luce Goodness-of-fit

Plackett-Luce can be used to derive a type of goodness-of-fit metric. This can gives us a score between 0 and 1 that says how "good" a predictor is for horse racing. The idea is that we make a comparison between a trained predictor and a "predictor" that only guesses randomly. This type of goodness-of-fit was used by both Benter [4] and Chapman [10] when they judged their own betting systems. The formulation can be expressed as follows (with R as the set of races and S as a predictor function and 0 as a random predictor):

$$\hat{R}^2 = 1 - \frac{\log(\prod_{i=1}^{n} \mathrm{PL}(R_i, S))}{\log(\prod_{i=1}^{n} \mathrm{PL}(R_i, 0))} \tag{6}$$

$$0(H_x) = \frac{1}{|R|}, H_x \in R$$

In this thesis we will be using this $\hat{R}^2$ value quite often and whenever we say "goodness-of-fit" we refer to goodness-of-fit as computed by equation 6. The value is typically between 0 and 1 but it can be negative for particularly bad predictors. It can be interpreted as how "good" the model is compared to

random model and can be used to compare different models in a more meaningful way than the other metrics we have discussed so far.

For example, BSP gets approximately 0.18 for $\hat{R}^2$ if the truncated P-L is used to depth 1. The starting prices are a good benchmark to compare against when we build new predictive systems. Table 3 shows P-L at depth 1 and at full depth for BSP, ISP and Forecast prices. Figure 2 shows goodness-of-fit at different depths. Rather curiously BSP goodness-of-fit drops below both ISP and Forecast prices at the end of tail. This does not mean BSP is a bad predictor but it does mean it may have more noise at the tail (i.e. terrible, hopeless horses have more noise) that penalizes it heavily compared to ISP and Forecast prices. It can also mean the assumption that Plackett-Luce loss makes is false to a large extent (the idea that if we remove any horses from the race it will not affect the relative probabilities of the remaining horses, violation of Luce axiom). Another cause may be by Betfair exchange mechanics that are specific to BSP prices: bad runners do not get as much betting activity and unsaturated market causes highly volatile betting odds and highly noisy predictions for terrible horses.

Despite that, P-L has some nice properties: it is differentiable and when used with linear regression it is convex. It is well-suited to gradient-based optimization and it can be computed in linear time. Unlike some many other loss functions listed here, it has a justified probabilistic interpretation. It also has a precedent of successful past applications: Xia et al. [24] compared different smooth ranking functions from learning-to-rank literature and concluded Plackett-Luce combines the best of consistency, soundness, differentiability, convexity and computational efficiency. The loss function used by Xia et al. in their ranking benchmarks is very similar to ours in equation 4, with slight differences in choosing $S(H_i)$ component.

The final advantage of P-L over simple metrics such as winner-predicted-correctly-accuracy is that it can extract more information out of small sample sizes. If we have a small sample size, e.g. 50 races with 10 horses each and we used simple win-loss labels, we would have 500 training examples with 50 positive labels. If we used P-L to a higher depth, not just considering the winner of the race, we can extract much more information out of each race; every permutation matters rather than just who won the race. This is particularly important in the fight against overfitting; by increasing the depth of P-L loss, our models are less prone to overfit.

Benter [4] used P-L in his betting system that was computed up to depth 2. As P-L has had good success in previous research and has many desirable properties, including a goodness-of-fit metric, we picked P-L goodness-of-fit as the primary tool for measuring and comparing different predictors of horse races. We usually only compute it to depth 1, as it is clear there is noise in the tail of P-L loss function for horse races.

3.6 Data sources

This section describes the data sources we used for all experiments and analysis in this thesis. We will also briefly describe how we stored the data for our own

Price	Goodness-of-fit, depth 1	Goodness-of-fit, full depth
BSP	0.1789	0.009199
ISP	0.1733	0.04544
Forecast price	0.1097	0.02569

Table 3: Plackett-Luce goodness-of-fit at depth 1 for BSP, ISP and Forecast prices. Measuring the goodness-of-fit of public betting odds is a good benchmark in comparing predictive systems.

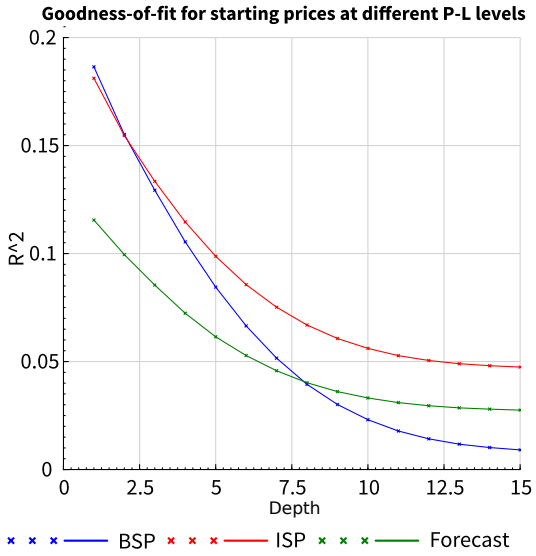

Figure 2: Plackett-Luce goodness-of-fit at different depths, for BSP, ISP and Forecast prices.

```
$ ./fetch_updates.sh historic
$ ./fetch_updates.sh daily
```

Table 4: Invoking Betwise data update scripts

experiments. At the end of this section you should have an understanding where the data comes from and what our data sets contain.

There are two entities we use as source of data: first is betwise.co.uk and the second is Betfair, which we will discuss in next two subsections.

3.6.1 Betwise

We will discuss betwise.co.uk first, which we will refer to as *Betwise*.

The main business of Betwise, at least today at 28 March 2018, is to offer a data set that they market as "Smartform Racing Database". They offer a subscription to their website for a monthly fee, and they provide a data package with historical information about in which order did horse races finish, who was part of those races, which jockeys were riding which horses etc.. Their highest quality data is on races located in Republic of Ireland and Great Britain. As of writing of this, their fees are listed as 45 British pounds for each month and 135 British pounds for all historical data. We paid Betwise for the duration of this research to obtain the data to research on.

Another major, crucial feature that they offer is that they do not only provide historical data; they also provide information on *upcoming* horse races, several days in advance. The data they give on future races obviously does not contain information that you only know post-race, such as finish positions but this feature makes it *much* easier to use their data for betting systems that make predictions on upcoming races.

The actual tangible format of the race data they offer is in the form of MySQL database dump files. The database tables they provide are called historic_runners, historic_races, daily_runners and daily_races. Each row in historic_races and daily_races contains information about some race, such as when it is going to be held, where is it located, what is the class of the race (class number indicates quality), material of the race track and other miscellaneous information. Each row in historic_runners and daily_runners contains information about a horse, for each race. Full information on the contents of the Betwise data set is too long to present here but documentation can be fetched from `https://www.betwise.co.uk/smartform_database_fields.pdf`[3]. The update procedure of Betwise data is performed by a rather simple execution of two shell scripts as seen in table 4.

We preferred PostgreSQL over MySQL because our Betfair data was already in this form and we wanted everything conveniently in just one type of database.

[3]We fetched this on March 28 and our version has a SHA-1 of 805c625beaea3a8bd2e8d595528f25700095be84.

```
#!/usr/bin/env bash
# This file is sync_sqlite3.sh
sqlite3 horsedata.sqlite3 <<!
.mode csv
.output daily_runners.csv
SELECT * FROM daily_runners WHERE
  foaling_date != '0000-00-00' AND
  foaling_date != '2001-01-00';
!
psql --user horse --dbname horse -f syncscript.sql

-- This file is syncscript.sql
BEGIN TRANSACTION;
DELETE FROM daily_runners;
\copy daily_runners FROM 'daily_runners.csv' WITH (FORMAT 'csv');
COMMIT;
```

Table 5: Converting data from Sqlite3 to PostgreSQL.

However, turning the Betwise data into PostgreSQL compatible format from MySQL was not entirely straightforward; there were incompatibilities, in particular with the way dates are presented between MySQL and PostgreSQL. The eventual solution we settled on is a bit roundabout: we wrote an update script that first updates an *Sqlite3* database (which was more accepting of MySQL -formatted dumps) and then updated the data in PostgreSQL by dumping the data from Sqlite3 and loading it to PostgreSQL. Table 5 shows an excerpt from out scripts that we use, with just one table, daily_runners.

We can run our update script daily and get all Betwise data in our PostgreSQL database.

Betwise data set provides the input data for almost all the features we engineered in our feature analysis section 4.3.1.

3.6.2 Betfair

Betfair needs a bit of introduction. They are a betting company based in United Kingdom and they are mostly known for their betting exchange. This is an online system where anyone can sign up and offer and take bets.

Betfair has a concept of back bets and lay bets. Back bets bet that a horse will win; you will receive a reward if you bet on the winner. Lay bets are the opposite side of a bet; you will receive a reward if your bet is on a loser. Betfair matches back and lay bets together to form a betting exchange.

Betfair markets itself as "fair"; you will get the most advantageous odds possible because Betfair does not build a margin for profit in the betting odds, like traditional bookmakers. Betfair instead takes a small commission out of all winnings and its exact size depends on your local country, the horse race in

question and their complicated discount system that encourages betting often. The commission is usually in the ballpark of 5%. You do not pay commission for lost bets, but since this is an exchange all lost bets go to the other party in the matched bet which means someone won some money and that money potentially pays Betfair some commission.

Betfair is a good source of data for horse races but it is of different nature than Betwise data. There are two different types of data we were able to obtain out of Betfair: historical Betfair Starting Prices (BSP) and trading data for some months in 2016 and 2017.

The Betfair Starting Prices (we mostly refer to them by their acronym BSP) are the betting odds at the time the race begins, on Betfair. The idea of a *starting price* is that you can place a bet but you do not know what your betting odds are going to be until the race begins. If you bet €5, your winnings, if your horse wins, could be €12 or €3; you do not really know; it will be decided by the market once the race begins. The marketing gimmick is that this ensures you get the "fairest" price, the price as it should be at the start of the race. This does not mean BSP is a good way to make money, all it means is that these are the smartest market prices we can get (assuming we believe in efficient market hypothesis). To make money our of BSP prices, the betting system has to be *smarter* than BSP prices and that means having to beat the market; a formidable task. If you believe in efficient market hypothesis, the BSP prices for each horse represent a very sophisticated estimation by the betting public that a horse will win the race. Even if you do not fully believe in efficient market hypothesis, figure 6 in a later section 3.7.3 shows that market gets smarter quickly at the last hour before the horse race begins.

On the other hand, the BSP prices are a good comparison point to judge betting systems. If it turns out some betting system is better at predicting horse race results than BSP prices, that system likely has the capability of making large amounts of money by betting against them. This is the main use of BSP prices in this thesis; we use them to measure how far behind our betting system is in terms of its predictive performance by comparing it against the BSP prices (as we alluded already in our methodology section in part 3.8).

As of writing of this, the BSP prices are freely available as CSV files at `http://www.betfairpromo.com/betfairsp/prices/`. We wrote a custom script that downloads them all and synchronizes our PostgreSQL database with new starting prices as they become available. We also wrote a custom Haskell program that can join the BSP price data with the Betwise data based on horse and jockey names. Almost all horse names are unique so this join matches quite well; there were mostly trouble dealing with some alternative spellings of horse and jockey names: for example "L'Marcy" may be missing apostrophe in the other data set and just be called "L Marcy". Even in these cases, fuzzy matching based on Levenshtein distance [19] achieved almost perfect matching.

The second data set that comes from Betfair is a trading data set. We collected this data by polling Betfair continuously for a few months between 8 September 2016 to 1 February 2017. As such, the time range it covers is short but it still covers almost 2000 horse races, which should be sufficient for many

types of analysis. This data contains the betting odds on the public market one hour before the race begins, at a granularity of a few seconds. We have this data for most horse races in United Kingdom and Republic of Ireland. The main use of the trading data set in this thesis is to study how smart betting markets are before a race and how beneficial would it be to bet as early as possible. These are analyzed in section 3.7.3 where we find that it is clear that markets get smarter the close we get to the start of a horse race.

3.7 Prices at betting markets

In previous section 3.6 about data sources, we talked a bit about BSP prices; the Betfair Starting Price. To recap, BSP prices are the betting odds that are in effect when the race starts and it is computed by Betfair based on currently open back and lay betting offers on the market. BSP prices can be seen as a sophisticated estimation by the betting public that a horse will win.

However, BSP prices are not the only public betting odds contained within our data sets. The Betwise data set contains two other prices: the Industry Starting Price (we use mostly the acronym ISP to refer to them in this thesis) and Forecast prices. This section is focused on analyzing starting prices and markets to scan for any quirks or unexpected behavior in them. Beating the starting prices in a prediction task implies it is possible to make large amounts of money through betting so it is a good idea to spend some effort to analyze them.

3.7.1 Industry Starting Price

The name "Industry Starting Price" in this thesis comes from a description as given by Betwise. ISP prices are starting prices determined by a collection of traditional bookmakers when a horse race begins. ISP prices are an alternative to BSP prices; instead of being decided by an exchange, they are decided by bookmakers. Both the Betfair exchange and bookmakers should be subject to efficient market hypothesis so our expectation is that these prices are similar.

One aspect that we could expect to be different between BSP and ISP prices is that the ISP prices could have profit margin built in. Bookmakers do not want to lose money to bettors so you would expect them to bake a margin inside their prices to make sure they will not easily lose money. And indeed, as we will see in this section, there is an easily identifiable margin baked in the ISP prices that biases them towards making a profit for the bookmakers. It is also possible to compute "readjusted" prices that extract the true opinion of bookmakers that some horse is going to win a race.

To start off this analysis, we will first plot a calibration plot that compares the ISP prices to actual, observed win rates. In figure 3 we have plotted the BSP and ISP prices against the true probability of winning at each price. These graphs are plotted by taking every horse in our data set, checking what is the ISP or BSP price for that horse and bucketing it in one of 50 bins based on the starting price. Then, the number of horses that won their race in each bucket

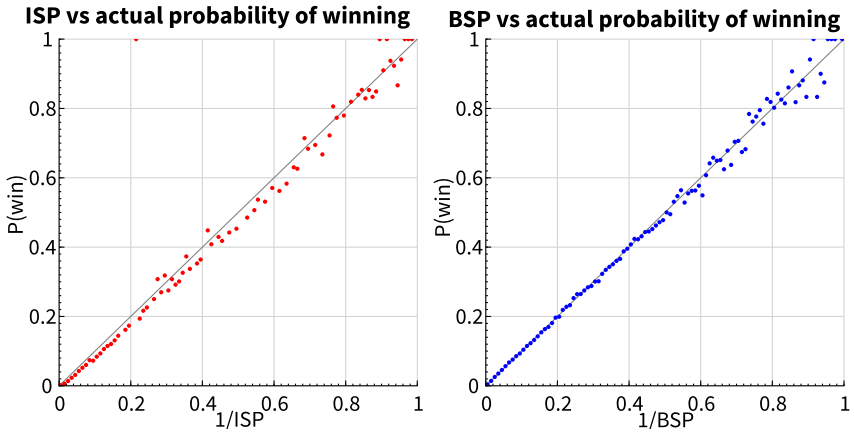

Figure 3: ISP and BSP calibration curves.

is divided by the total number of horses in each bin. This gives us an empirical estimation of the true winning rate in each bin. Finally, we plot a point for each bin, where x-axis has the implied probability of win by the ISP price and y-axis has the empirically computed probability. A well-calibrated set of predictions should fall on a straight line from lower-left corner to top-right corner.

If we study this figure, it can be seen that the points for ISP prices do not exactly line up with the straight diagonal line. The empirical probabilities are slightly lower than implied by the ISP price. The implication is that randomly betting on ISP price will quickly drain all your money (whereas with perfectly "fair" pricing your the expected value of your winnings would be zero, not negative). This gap between fair prices and actual prices (i.e. vertical distance between the diagonal line on figure 3 and the actual points) identifies the profit margin the bookmakers have put in and proves that the bookmakers do not solely rely on being smarter than bettors to get their profit; they rely on the combination of smarts and this margin.

The situation appears to be different on Betfair's side and its BSP prices. Betfair's business model is about taking a small commission out of winnings on their betting exchange, not about offering biased odds with a profit margin baked into them. It is not that Betfair is being more charitable: Betfair still gets its money; most winnings will pay some commission. However, in terms of observed betting odds, BSP is clearly more in line with actual observed probabilities as we see in figure 3.

There is a trick here that can be used to readjust the ISP prices to extract the true opinion of bookmakers and it is called model calibration. The high-level idea is that we plot a line through the points in figure 3 and use it to pull the predictions to the straight line. For example, if ISP price implies that probability of winning is 20% but actual observed probability of winning is only 15%, we know that for every prediction (as given by ISP prices) around 20%

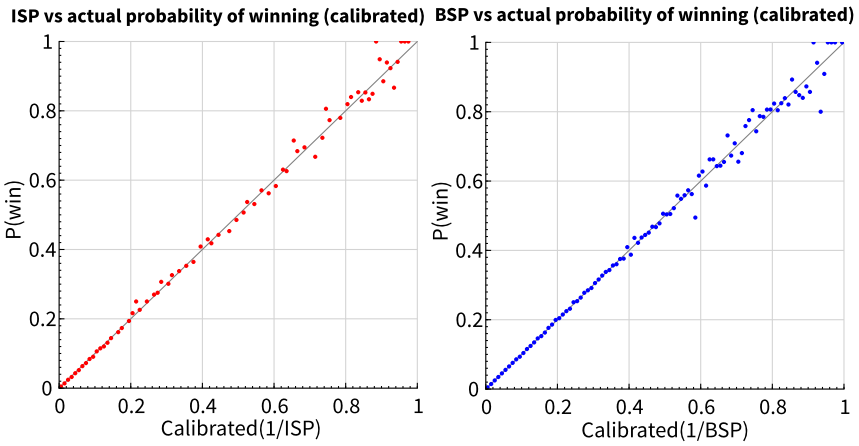

Figure 4: ISP and BSP prices after calibration.

should be adjusted by 5% to pull it to 15%.

A good algorithm to do this utilizes a specialized type of regression: the *isotonic regression*. A good introduction to isotonic regression is by Barlow and Brunk [3]. Isotonic regression can fit a free-form line to a set of points, minimizing the least-squares loss between the line and the points. What makes isotonic regression special is that it is restricted to being monotonically increasing and that it can be fitted in linear time to the number of points. The points on a calibration curve are mostly increasing, with some measurement noise. As long as there is enough data to compute good estimations of empirical probabilities for each bin, isotonic regression works well and tolerates noise in measurements.

The bins for isotonic regression in the following experiment are computed with the constraint that each bin is (almost) equally sized in terms of how many runners are in each bin. There are a lot more data points for horses that have a low probability of winning than for horses with high probability of winning; most horses are *losers*.

The calibration makes it easier to compare if bookmakers produce smarter prices through calibrated ISP prices compared to BSP prices. Figure 4 is the same as figure 3 but the ISP prices that are used went through calibration before being plotted. After calibration, ISP prices are much closer on the diagonal line on the calibration plot compared to uncalibrated prices. Also, table 6 shows goodness-of-fit $\hat{R}^2$ P-L score at depth 1 for BSP and ISP prices, before and after calibration. We can see that BSP prices are barely affected by calibration; in fact, BSP prices slightly suffer from calibration. ISP prices are more strongly affected. The difference is not large; the Plackett-Luce goodness-of-fit calculation normalizes probabilities before it is calculated, which already does some of the work calibration. Overall, ISP prices have a slightly lower goodness-of-fit than BSP prices.

Price	Plackett-Luce Goodness-of-fit to depth 1
BSP	0.1789
BSP (calibrated)	0.1787
ISP	0.1733
ISP (calibrated)	0.1768

Table 6: Goodness-of-fit before and after calibration.

The conclusion that can be drawn from this analysis and calibration is that ISP represents a sophisticated estimation that horses will win but there is a bias baked inside them for the benefit of bookmakers. BSP prices do not display the same quirk. Overall, BSP and ISP prices are very similar after calibration, with BSP being a slightly better predictor. It is unlikely that betting against ISP prices is wise unless a betting system is able to produce substantially better predictions than ISP prices, to overcome the profit margin built in them.

3.7.2 Forecast prices

Calling forecast prices "prices" is a somewhat of a misnomer because these are not prices you can actually bet at. The forecast prices come from Betwise data set and they are available several days before a horse race begins. Just like ISP and BSP prices, they can be interpreted as an estimation that a horse will win the race. It is unclear where the Forecast price actually comes from; it is only described as "the forecast price from racecard" in Betwise documentation. Racecards are information records, often a booklet that contains information about horses in a horse race, such as jockey name, time of race, who is the favorite, prize money etc.

Forecast prices are particularly attractive because they are available early and can be used as a feature in a machine learning model for horse racing. In a later section, where we perform feature engineering, we can see from table 9 that the forecast price by itself is one of the strongest features available that is available and usable before the race begins.

In figure 5 on the left forecast price recall curve is plotted to see if they are biased just like we did for ISP prices in previous section. It can be seen that forecast price also shows some bias. There is one quirk in Forecast prices that are not present in either BSP or ISP prices: they do not take non-runners into account. Non-runners are competitors in a horse race that declare, some time before the race begins, that they will not participate. This can happen at any time, even just minutes before the race begins. The starting prices like BSP and ISP are decided at the start of a race so they do not need to care about non-runners; if a horse has a starting price, they participated in the race. However, forecast prices, in general, assume all competitors are going to race several days before the race begins. For our calibration plot, the Forecast prices are normalized by removing non-runners from the race and adjusting the remaining win probabilities for each runner to sum up to 1.

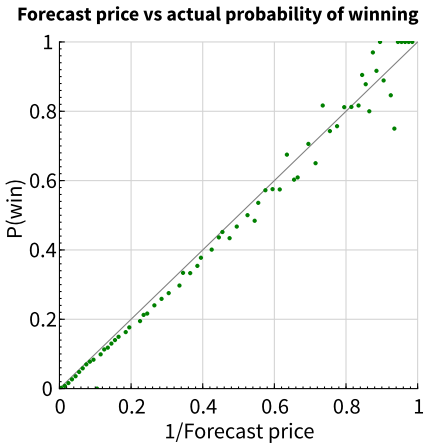

Figure 5: Calibration plot of Forecast prices. The Forecast prices show a slight bias towards overestimating the probability of winning.

Non-runners in general add annoyances to analysis efforts. BSP and ISP prices only exist for horses that actually decided to participate in a race; forecast prices exist for every horse.

3.7.3 Prices before the race begins

All the prices discussed so far are either starting prices (BSP and ISP) or a fixed price given at a set time (Forecast price). However, our data set contains information on betting odds an hour before each race for a few thousand races between September 2016 and February 2017 (see section 3.6 on our data sources). What makes this data interesting is that it allows us to study if markets get smarter as we get closer to the start of the race in time.

In figure 6 the P-L goodness-of-fit at depth 1 (see equation 6) is computed for each race in this data set and averaged. The plot was rendered by taking each 30-second point between 0 and 3600 seconds before the scheduled time the race begins and computing goodness-of-fit for each point. The goodness-of-fit of BSP is plotted as a horizontal line near the top of the figure. Finally, there is a black line is drawn using isotonic regression over mid-market rate. We also plot trading volume, with y-axis showing geometric mean of traded volume on each horse race market in our data set at that point in time.

It indeed can be seen that market gets smarter during the last hour before the race. This particular result is not surprising; more refined information comes in to the market as public can observe the horses before the race begins. There is also ample evidence in research focusing specifically on Betfair trading that confirms information enters the market very quickly. A good example is by Croxson and Reade [12] who studied betting activity in soccer games. Quoting from their conclusion: "Our findings suggest that prices swiftly and

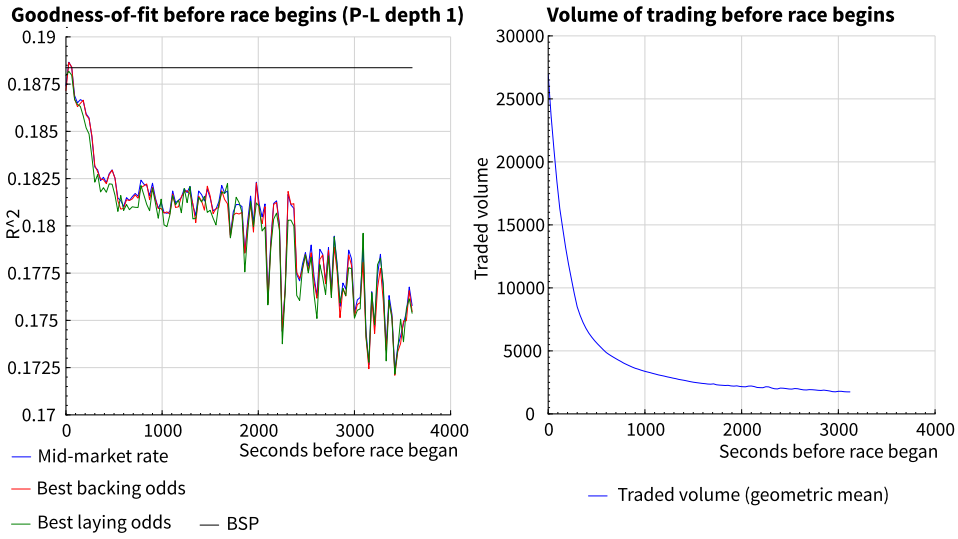

Figure 6: Goodness-of-fit of market prices one hour before race begin, using P-L at depth 1, along with trading volume

fully impound news.". This is likely also true for horse races.

There appears to be significant uptick in goodness-of-fit in the last 5 minutes. This could be caused by increasing betting trading activity or some particularly useful information that tends to enter the market only at the last minutes.

This information suggests that if you want to bet against the market; depending on how the betting system is set up, it can be beneficial to bet as early as possible so that the competition on the market is least sophisticated as possible. The system should bet at a time the last information becomes available that the system is equipped to utilize. For example, if the betting system only uses historical horse winning data to predict winners, the bets should be placed immediately when betting becomes possible on the horse race because there is no new information coming in that the betting system makes use of so there is no point in betting late. However, in our experience, Betfair betting markets often have very low volume of bettors days before the race so it is also possible no one will accept bets at very early stages. A data set that contains trading data for further back than 1 hour before races begin would be useful for researching the sophistication of market at such times. Betfair sells this data but for us it would be prohibitively expensive to buy in large quantities.

3.7.4 Concluding remarks of of starting prices

In summary, while BSP and ISP are both good predictors of race results, they have slightly different characteristics. Most importantly, ISP prices are biased towards making a profit for bookmakers. For this reason, it is likely a bad

idea to bet against ISP prices, unless equipped with a very powerful betting system that can overcome this bias. For analysis purposes, fortunately, it is somewhat easy to extract a fairly good estimation of bookmakers' true opinion of the probability that a horse will win, by calibrating their prices. Forecast prices are not quite as good estimators as BSP and ISP prices but they have the advantage that it is possible to use them as a feature in a betting system model, because forecast prices are available early.

3.8 Summary of methodology

There are various ways to judge the performance of a model that predicts horse race results with various trade-offs. We chose Plackett-Luce for its desirable properties but even P-L has issues when it is computed to full depth, as we saw in figure 2. P-L is a loss function and it can be used with any model that outputs some unbounded numerical value for each horse in a race, such as linear regression or a neural network with a linear output layer.

The primary way we will judge our models is by using goodness-of-fit metric derived from Plackett-Luce loss at depth 1. Public starting prices such as BSP and ISP on the market are a good way to establish a benchmark to compare against. If some model attains a substantially better goodness-of-fit than either BSP or ISP, it likely implies that a profitable betting model can be built.

All numbers where we judge the performance of a model will be computed either by 10-fold cross-validation or a 80%/20% training set/validation set split, in order to obtain an estimation of performance on out-of-sample data. We use cross-validation for cheaply optimized models that do not take long to train. Some models take hours to train on our system and we forego cross-validation in favor of a simple training set/validation set split for these tasks.

Our data sources originate from Betwise, a company that specializes in providing horse race data in exchange for a fee, and Betfair, a betting exchange company. We can use starting prices such as BSP and ISP as benchmarks when we judge our own models.

4 Feature analysis

We start this chapter with an illustrating example. Three horses are about to race in a horse race. They are racing on Chester, a prominent race course located in England. One of the horses is being ridden by an experienced jockey. Another horse has a good track record but fell in a race two months ago, sending the jockey to hospital; it is unclear if this horse's performance is going to be the same as before. The third horse is mostly average in every respect but is wearing a tongue-tie; a controversial piece of equipment placed inside the horse's mouth to prevent the tongue from going over the bit for the purpose of keeping the horse under control.

Given all this information, which horse would you expect to win and at which probability? There are many different factors that can be made use of, with our

example alluding to some of them. Our Betwise data set has large amounts of information on historical races and horses that can be made use of.

The goal of this chapter is to gain an understanding which factors of a horse race explain the outcomes best. We will see that best predictive features are those that take past performance into account and that some factors that might intuitively seem important are almost useless. We start by validating two common factors used by punters: the draw bias and pedigree.

4.1 Draw bias and race course properties

Draw bias is an example of a widely discussed factor in horse racing, with several websites systematically listing draw bias for different race courses. In a horse race, each horse starts in a randomly drawn stall. The stall number determines if horses start on the left or right of a race course. Most race courses have twists and turns. We can surmise that, given a certain shape of a race course, different stall numbers may be more advantageous or disadvantageous to the horse who starts in a certain stall.

This brings an interesting question. How do the starting conditions on a race course affect the horse race results, if we ignore any innate properties of the horses themselves? Aside from stall number, the race could be affected by the weather or distance. Do race course -specific circumstances have a very strong effect on results and do they differ between race courses? The answers to these questions can guide how much research effort should be performed on analyzing the race courses themselves.

We could not find formal research that studies draw bias in detail. We did find websites around the Internet that systematically list draw bias for different courses. There is an aptly named website at `http://www.drawbias.com/`[4] that lists draw bias for most courses in Great Britain and Ireland for different distances, along with a short, written analysis of draw bias for each course. However, the analysis is somewhat crude; the tables they show for each course is a very simple tabulation with a simple tally of winning for top and bottom 50% of stalls. Despite its crudeness, this is still useful and highly interpretable. While we want to answer a subtly different question than this website (mainly, how much do course circumstances affect horse race results), the website can guide as what features to consider in our analysis. For example, drawbias.com also has some analysis on different *going*[5]conditions and how it affects draw bias. We chose to use *going* in our own analysis as well.

Another aspect we can do better than this website is that we can consider full ranking of horses (using Plackett-Luce) competing on the track whereas drawbias.com only considers winners. This should give us much more reliable estimates of draw bias than you can do with just winners; we use much more data to draw our conclusions.

[4]We accessed this website at 2 April 2018.

[5]Race tracks have a "going". Going refers to the condition of the race track at the start of race. For example, "firm" going means the surface is firm and solid and "heavy" or "soft" often means the ground is wet and difficult to run on.

We will analyze draw bias and race course properties as follows:

1. Run this step and step 2 separately for every race course. Use bootstrap aggregation to train a small neural network model using the following features as inputs (a, b and c are "course features", d is the "horse feature"):
   (a) Normalized stall number (0 = horse starts at lowest numbered stall, 1 = horse starts at highest numbered stall, normalized to the number of runners in a race).
   (b) Normalized *going* value. Normalizing going is actually rather complicated for practical reasons; the way *going* is represented in our Betwise data set is not represented in convenient, machine-readable way; it is a human-readable description of going instead of something easily machine-readable. The full normalization procedure is in appendix A. The value of this feature is 0 on the worst going possible and 1 on the firmest going.
   (c) Standardized distance on the race course. Typically, race courses support races that are of different lengths. If the race is very long, it can mitigate (or exaggerate) draw bias.
   (d) Average past horse position.
2. Also train a model that only uses average past horse position as a feature.
3. Sort all race courses by a relative custom goodness-of-fit improvement achieved by the bagged model in step 1 compared to the trained model built in step 2.

This should extract race courses that are most strongly affected by race track conditions rather than any innate property of a horse or its jockey. We add the average past horse position as a feature so that we can examine if going has an effect; if we did not add this, we likely would not be able to gauge how going affects the results (because for each race, the input value for *going* is equal to every participating horse, we would not be able to differentiate them between competitors in each race). It may be that very bad going on a race equalizes differences between horses so that a strong horse does not have quite as much advantage in the race as it would have on a clear weather, very firm race track.

In step 1 we use bootstrap aggregation, a method proposed in a seminal paper by Breiman [7], that can reduce variance of machine learning models. In bootstrap aggregation, we take our race data and we sample from it N times, where N is the number of items of the training data, *with replacement.* The expected number of distinct samples in our bootstrap sample is $N(1 - \frac{1}{e})$, which is approximately 63.2% of the size of the original data set. The samples not chosen to the bootstrap sample can be used for out-of-bag estimations, to estimate how well our predictions would hold up for unseen data. Repeating bootstrap aggregation can produce good out-of-sample estimations with low measurement noise.

The reason for this choice is that we want our estimations to be as unbiased as possible to be more confident that they are reliable and not plagued by noise; an important factor to consider because some of the race courses have a very low number of races. Some of these race courses have less than 100 races in our data set and models trained on them likely easily succumb to overfitting. We are also helped by Plackett-Luce in that we consider full ranking, not just winners, to optimize; hopefully further decreasing variance.

The exact method of training is by using gradient descent with Adam heuristic (a type of momentum-based gradient descent method) [18], with mini-batches of 10 races each, optimizing for full P-L loss, and doing 16 separate trained instances of a model, all using different bootstrap aggregates. The results are judged by computing goodness-of-fit on out-of-bag samples. We choose a small neural network (just one hidden layer, with 3 neurons) to be able to model simple interactions and non-linearity between different features; something that linear or logistic regression would have harder time to do.

In step 3, we sort all race courses by goodness-of-fit; this should extract which race courses have most impact on runners disregarding the innate properties of the horses themselves. The metric we use is the relative value between goodness-of-fit between a model that only uses average horse position as a feature versus a model that uses the average horse position plus course-specific features.

The top 5 and bottom 5 race courses by goodness-of-fit are listed on table 7. The top race course is Sligo. Of these race courses, Chester is widely known to have very strong draw bias. Curiously, there are some race courses where using course features has a negative effect; although the negative effect is not quite strong as the positive effect is on the top courses. The worst course is Lingfield. We ran the training algorithm from start to finish a few times just to make sure that the results are stable; some minor differences happened with goodness-of-fit values but the difference in rankings was almost unaffected; possibly thanks to our bootstrap aggregation procedure.

There is no obvious pattern to why some courses are strongly affected. A country may indicate something. From the top 5 race courses, 3 are located in Ireland (Sligo, Ballinrobe and Wexford). The bottom 5 race courses are all in UK. It bears to note that in our data set, 92000 races were held somewhere UK and 25000 races are in Ireland; only about 20% of overall races that were part of this analysis were located in Ireland. This may indicate there is something fundamentally different between UK and Irish race courses but it is also possible this is simply a spurious correlation so we refrain from making conclusions based on this observation.

For the purposes of this thesis, we do not care deeply why the starting conditions on some courses have a strong effect on results and others do not. The most important observation is that for some race courses, the starting conditions can have a substantial effect on the outcome of a horse race. The average past horse position in a race is a rather strong predictor already; a model that can improve upon that by 30% or more in terms of goodness-of-fit just by using course-specific features is rather significant. Some future research could delve more deeply what makes the top courses different from the bottom

Course name	P-L horse feature only	P-L horse and course features	Difference %
Sligo	0.022	0.040	79%
Chester	0.018	0.026	42%
Ballinrobe	0.038	0.048	41%
Wexford	0.039	0.054	36%
Wetherby	0.033	0.046	35%
-	-	-	-
Bath	0.030	0.027	-4%
Leicester	0.028	0.026	-5%
Newmarket	0.018	0.017	-5%
Windsor	0.030	0.028	-7%
Lingfield	0.029	0.027	-8%

Table 7: Effect of course bias for top 5 and bottom 5 courses. "P-L horse feature only" refers to goodness-of-fit of a model trained only with the average past horse position. "P-L horse and course features" includes course-specific features. "Difference" is the relative difference between the goodness-of-fit values and the table is sorted according to this last column (with strongest course effects at the top of the table). 100% would mean the goodness-of-fit using course features improves goodness-of-fit to twice the value of a model that only considers average past horse position. 0% implies the course features are useless.

courses on this list.

Overall though, most race courses are not strongly affected. The average horse race position by itself is a much better predictor.

4.2 Pedigree

Race horses are often bred for competition. The owners of top-performing horses may rake in large amounts of money by letting them breed with another owner's horse to produce high-performing offspring. We can surmise that some characteristics of a horse are inherited and that the pedigree of a well-performing horse is likely to also perform relatively well. A cursory search on the Internet finds several websites dedicated to collecting and informing about the ancestral trees of horses for betting purposes. There is also formal research on horse genetics such as Binns et al. [5] who studied the effect of a certain gene on optimal racing distance for a horse. Clearly, the expectation is that the parentage of a horse is of importance. Our question is, is pedigree truly useful in predicting horse race results? Using our data sets, we can find out.

Some horses have very large pedigrees. In table 8 there is a list of top 5 horses by the size of their pedigree, for the horses that we have in our data set. The top horses have hundreds of descendants and likely have made their owners

Name	Number of immediate descendants	Top rank %
New Approach	335	99.96%
Mastercraftsman	308	98.79%
Zebedee	272	99.36%
Bushranger	268	89.78%
Equiano	262	93.15%

Table 8: Pedigree sizes of top 5 horses, along with their ranking among all horses. The ranking is computed with a sparse horse model we will describe in section 5.2.

significant sums of money. The top 3 can also be rightfully called the best of the best in racing horses, as they are in top 2% of all horses.

To analyze if the pedigree of a horse is successful in an intuitive way, we should rank horses in some way, to be able compare one set of horses to another. A simple way would be to consider some simple feature such as average finish position or official rating of a horse. The approach we take is that we compute a rank number for every horse and then compare the average rank of the pedigree for different horses and examine if they differ significantly.

In section 5.2 we will be building a sparse model to rank every horse. We will not go into detail how that model works right here in this section; suffice to say that it outputs a number for every horse, jockey and trainer. That number is their rank and it is comparable with every other horse (this would not be the case for something like average horse position; a horse that only competes in low quality races might only look good because it never had any quality competition; it would not be *comparable* with *all* other horses). This is how we computed the rank % in table 8.

We will examine the relation between the rank of the sire and it is pedigree. It is good to note here that while we have the names of all sire and dam[6] horses for some horse, we do not necessarily have any other data on them, if their race history predates 2003. For this analysis, we exclude parents that have less than 5 races in their track record that we have in our data set.

We find that the correlation coefficient between a sire and it is average pedigree rank is approximately 0.283. So indeed, there is some correlation here. To give some intuitive sense of these numbers; if correlation coefficient was almost 1 it would mean the offspring is almost squarely ranked in the same position as its parent in the ranking of all horses in our data set. If correlation coefficient was 0, it would mean there is no linear correlation between ranks of sire and its offspring and likely also would imply trying to select horses for breeding is meaningless. In that sense, 0.283 is a rather intuitive result; breeding matters but it is not a very strong effect. One caveat should also be considered: this result only considers descendants *that actually participated in horse races.* It is possible that these sires in reality have many more direct descendants but

[6]The sire refers to the male parent of a horse. The dam is the female parent of a horse.

they may not be racing; perhaps because of disappointing performance or other factors. For the purposes of predicting horse race results however, this should not be a problem for us. It can be a problem if the goal is to judge to what extent descendants inherit their abilities disregarding horse racing activity, but that question is not the focus of this thesis.

We can also compute the correlation coefficient between a dam and it is pedigree and we find that it is approximately 0.137. It is quite a bit smaller than the correlation with sires. This could have several reasons. One could be simply that dams have much smaller pedigrees due to biological reasons and our estimation has relatively more noise in it. Another reason could be that in breeding sires are considered much more important and the foals are trained according to the performance of its sire and the track record of the dam is considered unimportant for training. Nevertheless, both dam and sire performances affect the performance of their foal, although sire is more important.

But does this translate into good performance if we wanted to predict horse race results? Based on correlation coefficient we computed, the answer should be yes. But we would like to be able to compare how much effect the pedigree has compared to other features, such as the course features in previous section; we need a goodness-of-fit result rather than correlation coefficient. To study this, we train a P-L model where the only inputs to the system is the horses' dam and sire rank numbers. We judge the results by computing P-L goodness-of-fit against a random model, as we have done with draw bias in previous section. We only consider races where at least half of the competitors have information on their pedigree; this is only approximately 7000 races out of 110000.

There is another unique aspect to pedigree that can turn out to be useful. In a typical horse race prediction setting, you would want to consider the track record of the horse itself instead of digging into its ancestors. But sometimes there is no track record; every horse has a first time when they go racing. This can make considering pedigree information useful. To study this, we train another model but this time, for goodness-of-fit, we only judge horses that were racing for the first time in their races. In our Plackett-Luce loss, we do not penalize other other horses at all. The goal is to see if sire and dam performance can meaningfully predict the performance of a horse we know nothing else about, because they are racing for the very first time.

We find that the goodness-of-fit of this result is approximately 0.00045 for the case where we use full P-L loss. The goodness-of-fit for P-L loss that only considers first-timers is approximately 0.00146. This is a rather weak result, especially considering our small data set. For comparison, BSP prices have approximately 0.0092 goodness-of-fit for the same data at full P-L loss.

It may seem puzzling why the prediction performs so badly when there is clearly a correlation between sires and their descendants in terms of their full ranking. We can offer two possible explanations: 1) the ranking number in itself (that we used to measure correlation coefficient between parents and their descendants) is not a great predictor of results in the first place and 2) horses that race together tend to be similarly ranked already; you do not put the best-of-the-best horse racing against a novice. These two factors together could make

prediction tasks using pedigree information almost worthless.

For pedigree information, we conclude that it does not provide enough signal about horses to be useful in prediction tasks in a meaningful way. Despite this, you can find large amounts of resources on the Internet giving out tips how to utilize this information. One example: during our background research we read Nick Mordin's [21] book on betting and an entire chapter is devoted to studying pedigree and how to utilize it for betting. There may be other ways to use pedigree information but for the purposes of this thesis, this result tells us we should not give further consideration for pedigree information at this time.

4.3 Other features

There are many other features to consider. We will not be spending time digging deep into each of them but we give a short description below on each of them. We should address some ambiguity in terminology we are going to use here: when we talk about a "feature" we are referring to a single logical item or concept that we can extract from data. One feature can manifest as one or more "variable"s which are the actual, individual inputs to our model (e.g. if we have 32 "variables" and our model is a neural network, we would use exactly 32 input neurons in that neural network). To give an example what we mean by this: "average normalize horse position" is one feature. But it is, in fact, two variables: the normalized average horse position and an indicator variable that is 1 if there are no past horse races for this horse (and thus, no average horse position can be computed). We do feature analysis on feature level, not on variable level, as we want to focus analysis on using certain types of information and see low-level variables more as an implementation detail. In the listing below, most features are just one variable but sometimes they are more than one, most commonly when there are no past races we add an indicator variable indicating so.

4.3.1 List of features

Average normalized finish position This is the average position of the horse's previous results, where each past position is normalized to be between 0 and 1, with 0 meaning the horse came in last and 1 meaning the horse came in first. If the horse is racing for the first time (i.e. there are no past races for this horse), then an indicator variable is set to 1 and the average variable is set to 0. Therefore, this feature produces two variables as input, the average and an indicator if average exists. This feature was used in section 4.2 to test the effects of missing information.

Average normalized jockey finish position This is the normalized finish position for jockeys. It is computed in exactly the same way as average normalized finish position for horses but looking at jockey instead. Jockeys typically have dozens or hundreds of more races behind them than their horses.

Average normalized trainer finish position Same as average normalized jockey and horse positions, but for horse trainers.

Number of races so far for horse This is simply the number of races a horse has raced so far before the race.

Number of races so far for jockey This is the number of races jockey has raced before the race.

Number of races so far for trainer This is the number of races trainer has raced before the race. Established trainers typically have large amounts of horses that have raced for them.

Has-wins This is a simple binary variable that is 1 if the horse has ever won a race.

Weight This is the weight that a horse carries to the race. Large portion of horse races in United Kingdom and Ireland are called *handicap* races, where a handicapper assigns a weight to each horse. The idea is that the handicapper attempts to balance the weights just right so that the horses finish at the same time. Weight is also influenced by jockey weight; all jockeys are weighted before the race to check that they and their kit weigh the correct amount.

Relative weight This feature is otherwise the same as *Weight* but is relative to other horses in the race. The average weight is computed for the race and is then subtracted from the weight of the horse.

Relative weight times distance This is the relative weight but has been multiplied by distance. The idea is that the effect of extra weight compounds as the race gets longer and has stronger effect. This feature was used in Benter's system [4].

Jockey claim Inexperienced jockeys can "claim weight". In a race with professionals and amateurs, some weight can be taken off the jockey to make it easier for them to win the race against more experienced jockeys.

Age The horse's age, in number of days. There are many races that are age restricted so often this feature produces inputs that are very close to each other for each horse in a race.

Days since last race The number of days since the horse raced last time. There are various reasons why a horse might take a break from racing and it possibly tells something about its performance in future races. For example, if it has been a long time since the last race, it is possible the horse got injured and will not perform as well when it races again.

Forecast price These come from Betwise data set but it is unclear where exactly it comes from. It is described as "the forecast price from racecard in decimal format". The forecast prices can be described as a relatively sophisticated estimation that a horse will win and it becomes available a few days before the race begins. We already looked at the forecast price in section 3.7. The forecast price alone is the best single feature in this list of features in terms of how well it can predict horse race results.

The forecast price can be seen as a predicted probability between 0 and 1 that a horse will win the race, not counting non-runners. We take the logarithm of this probability instead of directly using the implied probability of forecast price. The reason for this is that this can cancel out exponential in our P-L loss function that uses an exponential to force all values to be positive. If you look at equation 4, all features end up in an exponential; using logarithm for forecast price makes it possible for the model to learn a "pass-through" for forecast price; theoretically it can optimize just to use forecast price itself as the output value, which is not possible if we do not use logarithm for it.

Race class This is an integer between 1 and 8 that describes the class of a race. In this context class refers to the quality of the race. A race of class 1 will have the best of the best racing on it and a race of class 8 is considerably lower quality. Irish races do not have classes and missing class is indicated with an indicator variable. This variable is given as-is in the Betwise data set.

Average normalized horse position weighted by race class This is the average normalized horse position but it is weighted by race class. The idea is that winning a class 8 race is much less noteworthy than winning a class 1 race. The weight is $9 - C$, where C is the class. This weighs a class 1 win 8 times higher than a class 8 win. Some races do not have classes and in those cases this variable becomes zero and a missing value indicator variable becomes 1.

Official rating This describes the "quality" of a horse and is decided by British Horseracing Authority. It often determines the amount of weight given for each horse to carry. Some horses do not have an official rating and an indicator variable is added for this case for the missing data.

Relative official rating Same as official rating but the mean official rating of all other horses in the race is subtracted from the official rating.

Number of horses in the race This is just the number of horses that are running in the race. This feature is equal to every horse in a race.

Competitor win rate This is a complicated feature that considers the win/loss statistics of the horses in the competition. The current horse is compared against other horses in terms of past races; if the horse lost to competition a loser count

is increased and if the horse win then a winner count is increased. The output variable is computed as subtraction between wins and losses. The intuitive idea behind this feature is to consider if this horse has won or lost against its competition in the past and summarizing that consideration into a feature.

Favorites This feature is derived from forecast price and breaks down to three variables. The first variable is 1 if the horse is a favorite for the race, i.e. the forecast price predicts highest probability of win for the horse. The second variable is the difference between forecast odds between first and second favorites but only if the current horse is a favorite, otherwise it is zero. The last parameter is distance, in forecast prices to the favorite. It is zero if the horse is a favorite. This feature should capture dynamics between strong and weak favorites in a race.

Recency weighted incidents Sometimes horses fall or have to be pulled over. Falling can cause injuries and horses sometimes get pulled over by the jockey if something bad is happening with the horse. The expectation in terms of predicting horse race results is that these incidents cause decreased performance in future races. This feature has one variable, which is zero if there have been no incidents. Otherwise it is the number of incidents weighted by how long ago they happened in number of days. Incidents that happened years ago will not matter as much as incidents that happened a week ago.

Long handicap This is the number many pounds of weight the horse is carrying over the official mark. It correlates heavily with relative weight feature.

Course features This feature is the effect of draw bias and other course features on the race. It is computed using an auxiliary model; in particular it is the same model we described in section 4.1 but now used as one of the features. The draw bias is trained separately and its output is used as a variable.

The number of progenies of a sire This is the number of progenies that the horse's sire has overall. We saw in section 4.2 on pedigree analysis that considering the pedigree of a horse is a very weak predictor in itself.

Handicapness This feature outputs 1 if the race is a handicap race and 0 if it is not. Handicap races have a handicapper that allocates weights to horses according to their ability.

Gender status This is a categorical feature and the genders are: colt, gelding, fillie, mare and stallion. As inputs it is presented as one-hot encoding using five variables, with 1 supplied for the corresponding gender and 0 for the rest of the inputs.

Bred country This is another categorical feature that describes the origin country of the horse. Although there are over 30 countries presented in the Betwise data set, the top 5 countries (Ireland, Great Britain, France, United States, Germany) account for almost all of them so this feature is presented as six inputs: one input for each country and one input in case the horse originates from any other country.

Gadgets Horses may wear gadgets such as cheek pieces, eye covers or tongue straps. Their presence may indicate past incidents that prompted their use. There are data points for six different gadgets: hood, visor, blinkers, eye shields, eye covers, cheek pieces, pacifiers and tongue straps. The data is sparse so all these gadgets are rolled into one variable, where the variable is set to 1 if any gadget present and 0 otherwise.

BSP We use Betfair starting price as a feature. We cannot actually use BSP at prediction time but it is useful as a comparison point, so this feature is omitted when we start estimating our performance on unseen data. Just like forecast price, we technically supply the logarithm of BSP because of the mechanics of our P-L loss function in equation 4.

4.3.2 Automatic feature analysis

We want to do feature analysis to determine which of all these features are actually useful in figuring out horse race results. We are going to use two different ways to do this, to make sure we will be able to catch any nuances just one method would not be able to do. First, we train a P-L model that only gets one feature from our list. We rank features according to P-L goodness-of-fit we get from each of the features. However, some features cannot be judged this way, e.g. the feature of how many horses there are in the race, because its value is the same for every horse and the model cannot be any better than a random model for such features. This is why we also perform a second, more complicated way to do feature analysis: we train a P-L model using a random subset of features. We train many models this way and each time take random subset of all our features. We assume that each feature contributes some unknown, but positive amount of goodness-of-fit to the final goodness-of-fit score. We optimize another model that tries to find these unknown contributions given the data.

By testing with enough random subsets of features, we should be able to obtain a ranking of features when they are being used with other features, rather than just their contribution when used by themselves. The algorithm we use to estimate contribution of each feature is described in algorithm 1.

This algorithm should provide an estimation how much do each feature contribute to final goodness-if-fit, when other features from our feature set are present. This scheme is far from being an accurate representation of how the features interact. For one, goodness-of-fit is not really additive; it is theoretically possible to get a result that implies certain combinations of features attain goodness-of-fit greater than 1, which is impossible. Highly correlated features

Algorithm 1 Feature analysis using random subsets of selected features.

1. Goal is to optimize vector G that contains estimations for contributed goodness-of-fit per feature. Repeat steps 2-5 until in step 5 G does not change meaningfully between iterations.
2. $R_i \leftarrow$ randomly sampled subset of all features at iteration i
3. $\hat{G}_i =$ goodness-of-fit computed from using features in R, obtained by optimizing a P-L loss with $S(H_i)$ as a linear regression model. (See equation 4).
4. Insert $\hat{G}_i$ and R_i in training set. Notate T as the training set so far and notate $\hat{G}_i$ as the ith label.
5. Using the latest training set built so far, optimize G by minimizing: $\sum_{i=1}^{|T|} ((\sum_{j \in R_i} G_j) - \hat{G}_i)^2$.
6. At the end: each G_i now contains an estimation of the goodness-of-fit it contributes to final model.

likely share their contribution and will look worse when compared to our simpler automatic feature analysis method that considers each feature individually. The expectation is that our list will rank certain features high that are not correlated with other features and the information they provide is useful.

We set the number of iterations on this algorithm to 500. At that point, we found that further iterations no longer significantly changed the estimation of vector G.

Feature analysis for the first method is listed on table 9. Feature analysis using the second method is on table 10. Starting with table 9, we can see that features that directly have something to do with past performance are the best features. For example, average jockey, trainer and horse position are all good features. Unsurprisingly, forecast price is the best single feature on its own. BSP wins but is not really usable in actual prediction tasks because it only becomes available after the race has begun.

Looking at mid-range, somewhat useful features have something that intuitively makes sense would affect results; such as weight carried or gender of the horse. Worst features are complicated, failed attempts at feature engineering such as the relative weight times distance or recency weighted incidents. Course features (listed as DrawBias) are not very useful compared to other features on table 9.

Looking at the second table, 10, we can see that the ranking has somewhat changed. BSP and Forecast price are still on top, although this time "Favorites" feature (see section 4.3.1 for description of this feature) went past it. Average horse, jockey and trainer position are now ranked much lower; although this could be because they are highly correlated and the credit they deserve gets divided between them. The new features in 10 that are not in 9 (because first

Feature	P-L goodness-of-fit, depth 1
ParentsAveragePosition	0.000006
RelativeWeightTimesDistance	0.000028
HasGadgets	0.000293
DrawBias	0.000334
RecencyWeightedIncidens	0.000401
BredCountry	0.000543
SireNumberOfChildren	0.000904
NumRacesSoFarHorse	0.001709
DistanceTravelled	0.001871
LongHandicap	0.002208
DaysSinceLastRace	0.002817
Age	0.003501
RaceType	0.003726
JockeyClaim	0.003765
Gender	0.004144
Weight	0.005325
NumRacesSoFarTrainer	0.005405
CompetitorWinRate	0.005822
RelativeWeight	0.006362
NumRacesSoFarJockey	0.006506
HasWins	0.006811
OfficialRating	0.012894
OfficialRatingRelative	0.018707
AverageJockeyPosition	0.022610
RaceClassAverageHorsePosition	0.030531
AverageTrainerPosition	0.032481
AverageHorsePosition	0.043337
Favorites	0.088669
ForecastPrice	0.105386
BSP	0.167711

Table 9: Feature analysis; one Plackett-Luce logistic regression model trained with each feature used by themselves. Features that yield the same inputs for all horses in a horse race are omitted from the table (e.g. “Going”).

Feature	Goodness-of-fit	Adjusted goodness-of-fit
SireNumberOfChildren	0.0016574	-0.00023800
Random	0.0017426	0
RaceClass	0.0020436	0.00084004
ParentsAveragePosition	0.0020882	0.00096474
Age	0.0020911	0.00097279
AverageTrainerPosition	0.0021827	0.0012285
TrackType	0.0023599	0.0017231
NumHorsesInRace	0.0024145	0.0018756
NumRacesSoFarTrainer	0.0024227	0.0018984
Going	0.0024229	0.0018990
RecencyWeightedIncidents	0.0024449	0.0019604
BredCountry	0.0024754	0.0020456
HasWins	0.0026311	0.0024803
DrawBias	0.0026452	0.0025197
NumRacesSoFarHorse	0.0027083	0.0026957
CompetitorWinRate	0.0027276	0.0027496
RelativeWeightTimesDistance	0.0028463	0.0030811
DistanceTravelled	0.0028774	0.0031678
LongHandicap	0.0028814	0.0031789
RaceType	0.0030993	0.0037874
AverageJockeyPosition	0.0031569	0.0039479
JockeyClaim	0.0032097	0.0040954
AverageHorsePosition	0.0033647	0.0045282
RelativeWeight	0.0034037	0.0046370
HasGadgets	0.0034365	0.0047286
Gender	0.0034706	0.0048237
Handicapness	0.0035227	0.0049692
OfficialRating	0.0036880	0.0054306
NumRacesSoFarJockey	0.0042743	0.0070675
Weight	0.0044163	0.0074638
DaysSinceLastRace	0.0044335	0.0075120
OfficialRatingRelative	0.0051188	0.0094249
RaceClassAverageHorsePosition	0.0063234	0.012789
ForecastPrice	0.018180	0.030833
Favorites	0.023120	0.081211
BSP	0.061829	0.167711

Table 10: Feature analysis; using algorithm 1 to obtain usefuless of features that by themselves would be useless. Compared to table 9, there some additional features we can analyze and are included in this table.

method cannot analyze them) are: RaceClass, TrackType, NumHorsesInRace, Going and Handicapness. Only Handicapness seems to rank relatively high (this feature is 1 if the race is a handicap race, 0 if it is not). This could indicate handicap races are quite different from non-handicap races and we should take that into account when we build betting models. In a handicap races, a handicapper places weights on each horse in an attempt to balance their differences. The goal is to make all horses finish at the same time. If the handicapper has skill, this could imply differences between horses are dampened and this is what makes handicap races different from non-handicap races. The relative ranking of Handicapness feature on our list indicates that taking this into account may be more useful than many other features such as the age of the horse, going in the race or jockey claim.

In order to make the listing on table 10 more comparable with table 9, we subtracted the goodness-of-fit obtained by random feature and set BSP goodness-of-fit at 0.167711 (from table 9) and normalized all other values to be some number between 0 and 0.167711. Overall, the values still are much more pessimistic in table 10 than in table 9. The presence of BSP price may pull all other estimations down; none of the other features seem useful when BSP already reflects all available information by efficient market hypothesis.

As a summary, the most important finding here is understanding what types of features are the best types of features. Aside from Forecast price, other useful features take past performance of a horse into account, such as average horse position or official rating.

5 Predicting horse race results

Previous sections have analyzed ranking functions and examined features that we could extract from our data sets. We pitted features against each other and estimated how well certain types of features predict horse races. The *theme* of previous sections has been about *understanding* horse races. Starting from this chapter, the theme changes from *understanding* races to *predicting and betting* on horse races. We want to build a system that can bet on horse races and profit from it.

How do we predict horse race results based on data? We will be using the Plackett-Luce loss with more sophisticated predictors than simple linear regression and we are going to make use of all the features we have engineered so far. In section 5.1 we will build a dense model. In section 5.2 we build a sparse horse model that has one optimizable weight for every horse, jockey and trainer. In section 5.3 we will also test a combined sparse and dense model and find that sparse model is less useful than we hoped.

5.1 Dense horse model

We first train a dense model. We take all the features we engineered in section 4 and put them in the same model. We use a simple neural network with 4

Model	P-L goodness-of-fit, depth 1
Neural network	0.14603
BSP	0.17833
Forecast Price	0.10972

Table 11: Results from training a neural network model on all our features. The goodness-of-fit for the neural network model is estimated from a 10-fold cross-validation procedure.

neurons in the hidden layer to make the model capable of learning some simple non-linear interactions between the variables. In particular, as we learned in our feature engineering section, features like handicapness are not effective if they cannot be combined with other features. We compute our $S(H_i)$ in equation 4 by a neural network, with one hidden layer with 4 neurons, using logistic (i.e. $\frac{1}{1+e^{-x}}$) activation function and with a linear output layer.

We use 10-fold cross validation to estimate out-of-sample performance. The loss function is Plackett-Luce and we judge this model with the same criteria we used for feature engineering: Plackett-Luce goodness-of-fit at depth 1.

Results are on table 11, along with BSP and Forecast price as comparison points.

The neural network is substantially better than just using Forecast price. It is, unfortunately, quite far from being as good as BSP prices. It is also not as good as prices one hour before the race (which we studied back in section 3.7.3). It is clear that even with all our features and a relatively sophisticated model, our predictions are not good enough to bet profitably. A more powerful model is needed.

5.2 Sparse horse model

All the models we have built so far have been dense. We did this for feature engineering and also for the model in the previous section. Most inputs in our models are populated; missing data is rare.

One example is the average horse position. As we saw in section 4, this is not a bad feature. But it does have some downsides. For example, a horse that races only in very low quality races does not have much competition anyway and it is much easier to win and get good results. If this horse is then bumped up several classes, it might fare much worse in higher quality competitions. Even though it looks like the horse achieved success in previous races, those wins matter less when the horse starts participating in higher quality races. Our simple average horse position feature cannot capture this nuance.

We have engineered dense features but this time we will try something a little different; a sparse model.

Instead of trying to compute the average horse position or trying to fiddle with alternative past position formulations like weighted average or class-

weighted average, we could instead consider adding a optimizable input for *every* horse. We can train a model that assigns a value for each horse that ranks their relative strengths. If optimized across all horses, this model should be unaffected by the problems with average past horse position because it always also takes into account the competition in each race.

For this we need to formally define a model. If w defines all optimizable parameters in our model, then w can be dissected like below:

$$w = \begin{cases} w_H & : \mathbb{R}^h \\ w_J & : \mathbb{R}^j \\ w_T & : \mathbb{R}^t \\ w_h & : \mathbb{R} \\ w_j & : \mathbb{R} \\ w_t & : \mathbb{R} \\ w_0 & : \mathbb{R} \end{cases} \tag{7}$$

These values are to be used with linear regression inside Plackett-Luce equation 8. Some explanation is in order. h, j and t are the number of horses, jockeys and trainers in our data set. For each horse, jockey and trainer we have one optimizable parameter, contained within vectors w_H, w_J and w_T. In addition, we have weights w_h, w_j and w_t, that tell how we should weigh the categories of horses, jockeys and trainers against each other. w_h, w_j and w_t weights are superfluous; not strictly necessary, as w_H, w_J, w_T weights can already express anything needed but we found that including them in our model makes the model converge to good results much faster. We will notate $w_H(H_i)$ to mean that we pick the value of the parameter for the horse H_i, with the same notation for jockeys and trainers respectively.

w defines the *parameters* of the model. As we will be using these with P-L loss function (see equation 4), the parameters are used with linear regression, with weights taken into account. Our predicting function, $S(H_i)$ function is defined below:

$$S(H_i) = w_0 + w_h w_H(H_i) + w_j w_J(H_i) + w_t w_T(H_i) \tag{8}$$

This outputs a "score", a real number, for every horse we run through it. A nice property of this model is that almost all the variables are interpretable. The weights w_h, w_j and w_t tell us how to weigh relative performances of horses, jockeys and trainers. w_H, w_J and w_T can be used to rank all horses, jockeys and trainers.

There are some problems that turn up if we try to train this model as-is. We will start with a demonstration. Using P-L and optimizing it without using truncation, we train the model exactly as we described it it equations 7 and 8. We split our data into 80/20 split for training set and validation set. [7]First

[7]We would have chosen 10-fold cross-validation instead if the optimization with our Haskell program did not take hours.

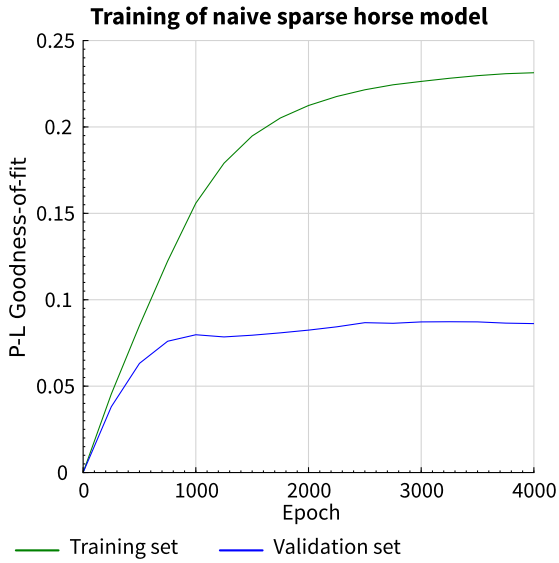

Figure 7: Training and validation set goodness-of-fit over training time.

problem comes when we encounter a horse, jockey or a trainer that is not in our training set. For example, the value of $w_H(H_i)$ may not exist because H_i was not part of our training set, but happens to be in the validation set. We work around this problem by adding another optimizable parameter, w_{zero}, that is used for any horse that is not in the data set.

Second problem is that it is easy to overfit this model. Figure 7 shows training score and validation score over time, with number of training epochs on x-axis. The goodness-of-fit is Plackett-Luce to depth 1. The validation set score plateaus rather quickly. There appears to be diminishing returns after around 1000 epochs of training. While we did use L2 regularization, for this particular problem the overfitting problem is likely inherent because the sample sizes are very small for each individual horse.

There is a third problem and this problem is about how we split our data set into training and validation sets. So far, in this thesis, when we have split our data into training and validation sets, it is done randomly on race-level. Some horse races are put into training set and some are put into validation set. However, this is not the best way to split our data into training and validation sets. We are testing our model on races that happened before the races we used in the training set. It could be said that for many races, we are doing our out-of-sample tests assuming a world where we run some races in the future, take a time machine, go to past and then test our models on some past races we did not account for in our training set.

A better, more sound way to do this is to split the training and validation set by time instead. We could choose all races between years 2010 and 2015 in

our training set and test on races in 2016 and 2017. There are unfortunately practical problems with this. Majority of jockeys and many horses race again in just few days after their previous race. Often jockeys race even on the same day. We cannot make a validation set that spans more than one day if we want our out-of-sample estimations to be accurate. This means training on years of data and testing on one day, then training again on years of data, adding one day to the training set, and testing on next day. For 2 years of validation set this means training our model over 500 times. Our current implementation of our training algorithm is too slow to make this feasible in short amount of time, especially in our later experiments where we will be combining our sparse model with dense features.

This problem is not unique to our sparse model; it also affects the other models we have trained in feature engineering and our dense model.

Overall, despite the overfitting problem, the goodness-of-fit obtained from validation set is relatively high, approximately 0.082212. This is better than most other features, it is only behind Favorites and Forecast price features in our feature engineering table 9. The sparse model is not quite as powerful as we hoped though; a more powerful model is needed if we want to make a profitable betting system. It is also weaker than our dense model.

Future work might use an alternative way to divide the data set into validation and training set or consider a different approach to introduce sparsity to our model. One idea would be to hash features into lower dimensional vector, which may fight overfit (Weinberger et al. [23]).

5.3 Combining sparse and dense model

The sparse model in previous section is quite good but we need to do better if we want a profitable system. The sparse model considers that every horse, jockey and trainer have some ranking number and that rank is used to predict the probability that they will win a race. However, various circumstances around a horse race can affect the probability that some horse will win in ways the plain sparse model cannot account for. For example we could have a situation where the horse is running on the Sligo race track (we saw in section 4.1 on course features that some race courses have special properties), where the stall number has significant effect on results. The stall number is random and independent of the horse itself. Instead of just using these ranking features, we should incorporate all our features that we listed in our feature engineering section 4.3.1 and combine our sparse model with them.

This is fortunately, rather straightforward. We take the w, the set of parameters in previous section and add all dense features to it. We will call this new set of parameters with both dense and sparse features $\tilde{w}$. The feature decomposition on equation 9 is otherwise equal to 7 except we added w_F as the set of dense feature to it.

$$\tilde{w} = \begin{cases} w_F & : \mathbb{R}^f \\ w_H & : \mathbb{R}^h \\ w_J & : \mathbb{R}^j \\ w_T & : \mathbb{R}^t \\ w_h & : \mathbb{R} \\ w_j & : \mathbb{R} \\ w_t & : \mathbb{R} \\ w_0 & : \mathbb{R} \end{cases} \tag{9}$$

The difference between equations 9 and 7 is the addition of w_F. This vector contains coefficients to be used in a linear regression for the dense features. Our new $S(H_i)$ function changes from 8; we need to add dense features to it. If the actual input values are defined by feature function $F(H_i)$ (that outputs a vector of input variables), then our adjusted formula is:

$$S(H_i) = w_0 + w_h w_H(H_i) + w_j w_J(H_i) + w_t w_T(H_i) + w_F \cdot F(H_i)$$

Where $\cdot$ denotes the dot product of weights and inputs.

Figure 8 shows the training and validation set errors on combined sparse and dense feature model. In this case, validation score can attain 0.1 goodness-of-fit between 1000 and 1500 epochs. The training was stopped at 2000 epochs as validation score started to drop.

The goodness-of-fit attained just with the sparse model was 0.082212. The goodness-of-fit attained with combined sparse and dense model is at 0.097482. While this is an improvement over just sparse model, this goodness-of-fit score is *worse* than just using Forecast price as a feature (we can see this if we check out table 9 in our feature engineering section, Forecast price attains goodness-of-fit 0.105386). It is also worse than dense model alone.

Does this mean sparse model is not worth it? Not exactly. While the performance is disappointing, we can still make some use of this model. We will treat the output of the sparse model as another feature we use in our dense model; i.e. we simply add it to our list of features. It still incorporates information that is difficult to capture with a dense model only.

To make sparse model better on its own, a clever strategy to split the data into validation and training sets needs to be found. Since each horse individually only has a few races to build a sample out of, it is very easy to overfit a sparse model that has an optimizable feature for all of the horses.

In next section we will be building a betting strategy and we incorporate combined sparse and dense model as one of the features.

6 Building a betting strategy

So far, we have been trying to build models to *predict* how horse race results are going to turn out. While it is possible to make rather good predictive systems

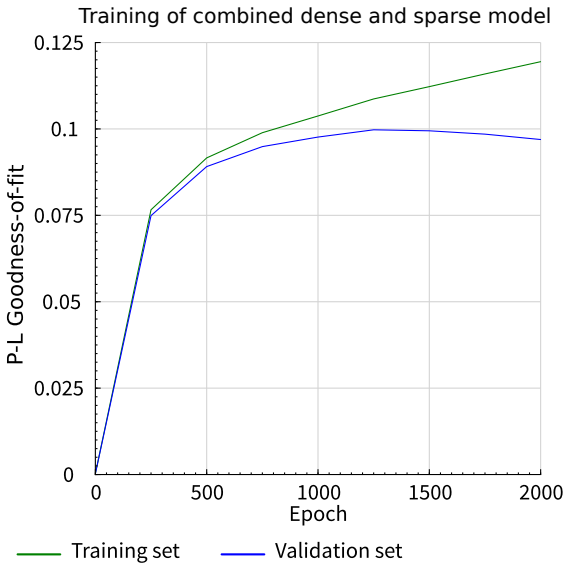

Figure 8: Training of combined sparse and dense model.

with enough feature engineering, in the end, our model cannot beat the existing market starting prices, BSP and ISP prices.

It might seem like this is the place to give up. How could we make a profitable system if our model is not smarter than the betting public? Fortunately for us, this does not mean that a profitable betting system is impossible. What we can do is *stack* predictive models for better predictions. This means stacking the betting public's current betting odds with our own model. If the resulting model is even slightly better than the already available odds, it can open an opportunity to bet profitably.

With this idea, we can envision a betting system with two different components. First component trains a predictive model that specializes in *improving* some existing estimation of horse race results. The second component, using the output from first component, decides how much to bet and when to bet. In this chapter, we first focus on building a predictive system that improves existing results before we move on to our end goal: building a profitable betting strategy.

6.1 Improving on starting prices

First of all, we need a probability that a horse will win or lose and our prediction should be as precise as possible. Unfortunately, as we have seen in our feature engineering and prediction results in sections 4 and 5, even our best model, so far, cannot get quite close to starting prices in terms of goodness-of-fit. If we tried to apply a betting strategy purely on these models we cannot expect

positive long-term returns from our strategy.

Does this mean we should pack up and give up? Not quite. As we alluded to in the introduction of this chapter, there is one avenue of optimizing we have not explored yet; we can build a model that specializes in *improving existing predictions, in particular starting prices.* So far, we have built models in a way that the final probabilities were produced from a vacuum; solely from our list of features that we engineered. However, this need not to be so. We can instead take the BSP prices as a base and stack our own predictions on top of them. We can optimize this from end-to-end, for minimized P-L loss. If we get even a small edge over BSP prices and can confirm the edge on out-of-sample tests, there is good chance we have produced a profitable model.

We reuse all methods we used in feature analysis and sparse models, the difference this time is in how we compute the final probabilities we judge against. Doing the feature analysis again on BSP stacked model can reveal interesting properties of BSP prices; the features we expect to be useful in a stacking model may be quite different from features that are useful for predicting from a vacuum. Below we have equation 4 reproduced. This is the formula that computes Plackett-Luce loss, and also describes how we obtain probabilities from our score function.

$$\mathrm{PL}(H) = \prod_{i=1}^{n} \frac{e^{S(H_i)}}{\sum_{k=i}^{n} e^{S(H_i)}} \tag{10}$$

We change this formulation a bit to include BSP prices. Let $\mathrm{BSP}(H_i)$ be the implied probability of a horse winning, by BSP price. We change our formula to this:

$$\mathrm{PL}(H) = \prod_{i=1}^{n} \frac{e^{S(H_i)+\log(\mathrm{BSP}(H_i))}}{\sum_{k=i}^{n} e^{S(H_i)+\log(\mathrm{BSP}(H_i))}} \tag{11}$$

We add the logarithm of BSP probability in the equation. If $S(H_i)$ is zero, the formula becomes simply BSP prices themselves.

We cannot use BSP prices themselves at betting time. But we expect that if a model can improve BSP prices, it likely also improves prices such as ISP prices or the prices before the race begins (we did a brief check on market prices before the race begins in section 3.7.3).

6.1.1 Feature analysis on stacking model

We start off with doing our feature engineering method we used in section 4.3.2 again but we stack the features on BSP prices this time, using equation 11. We train one linear regression model per feature and then sort all the features based on P-L loss to depth 1.

The results are in table 12. For one, this table is quite different from 9 in terms of ranking of features. The only feature that is clearly good in both feature analysis is the forecast price. Another surprisingly useful feature is the presence of gadgets; in our feature engineering section, gadgets were almost

useless in terms of predicting horse race results. Only 7 features *improved* on BSP, the rest resulted in a decrease of goodness-of-fit over BSP prices. There is one curious result: four features that are all about properties of the race itself are all in top features: going, race class, handicapness and number of horses in a race. We saw previously in our feature engineering section that handicapness was one of the features that alone is useless but useful when combined with other features. All these features yield identical inputs for each horse in a race. We ran the analyzer several times and the results seem fairly stable; with only very minor changes in how features ended up ranked. Overall, the differences between features are fairly small.

There are no truly curious findings in this feature analysis this time. One thing it tells us is that some features may be actively harmful if we try to put them in our stacking model, because they lower overall goodness-of-fit. We picked top 8 features from table 12 and used them for our stacking model. We also tried using PCA for this problem but it did not work out well. Table 13 contains PCA values we obtained using top principal components but we abandoned PCA quickly when we noticed that the results from just picking top features works better.

6.1.2 Optimizing stacking model

We used the top 8 features we engineered (i.e. all features that improved on BSP prices, from table 12, plus the out-of-sample predictions from our combined sparse and dense model) and chose a simple neural net as the model that computes $S(H_i)$ in equation 11. The neural net has one hidden layer with 4 neurons. The optimization target is Plackett-Luce model at depth 1. We used a simple random split of our data set; 80% for training and 20% for validation.

We also make use of calibration, utilizing isotonic regression, that we used on starting prices in section 3.7.1 as we noticed this significantly improved results. We computed calibrator on the training set and used it to calibrate the validation set.

Our neural net had approximately 50 optimizable parameters. We used L2 to add some regularization, with 0.0001 appearing to be a good choice for this particular model.

Unlike in earlier parts of this thesis, this time we judge our model based on the money it generates if we used it for betting. Before we go through our results, we will describe our betting strategy.

6.1.3 Kelly criterion and betting strategy

The betting strategy we built uses Kelly criterion as a sub-component. As we've mentioned in our background on information theory back in section 2.2, J. L. Kelly Jr., studying information theory, in 1956[16], came up with a formula that maximizes the logarithm of wealth over long term in a betting system. The most commonly cited version of the formula is as follows:

Feature	Goodness-of-fit
Favorites	0.178068
Gender	0.178136
LongHandicap	0.178164
BredCountry	0.178194
AverageJockeyPosition	0.178205
RaceClassAverageHorsePosition	0.178218
TrackType	0.178225
ParentsAveragePosition	0.178234
AverageHorsePosition	0.178236
JockeyClaim	0.178238
OfficialRatingRelative	0.178261
AverageTrainerPosition	0.178266
RaceType	0.178267
HasWins	0.178267
DaysSinceLastRace	0.178274
RecencyWeightedIncidents	0.178284
NumRacesSoFarHorse	0.178287
NumRacesSoFarJockey	0.178289
RelativeWeight	0.178292
NumRacesSoFarTrainer	0.178293
Age	0.178303
OfficialRating	0.178304
DistanceTravelled	0.178304
SireNumberOfChildren	0.178308
RelativeWeightTimesDistance	0.178315
Weight	0.178319
Going	0.178336
RaceClass	0.178337
Handicapness	0.178338
NumHorsesInRace	0.178338
CompetitorWinRate	0.178343
HasGadgets	0.178349
ForecastPrice	0.178362

Table 12: Feature analysis on a model that stacks on top of BSP prices. For this data set, BSP price goodness-of-fit is at 0.17833; only features from Going and below improve on it. Any goodness-of-fit value in the table larger than that has been able to slightly improve the predictions.

P-L goodness of fit at depth 1	Number of principal components
0.17777	1
0.15979	2
0.16871	3
0.17375	4
0.17727	5
0.17795	6
0.17848	7
0.17867	8
0.17866	9
0.17864	10
0.17872	11
0.17876	12

Table 13: Goodness-of-fit in validation set; obtained by training a linear regression P-L model using all features, after PCA has been applied to those features and we have picked only top N principal components. BSP price goodness-of-fit is 0.17833; this is met at 7 principal components. The goodness-of-fit is bad at only few principal components which may suggest PCA is not a good method to process our data set.

$$f = \frac{bP(\text{win}) - P(\text{lost})}{b} \tag{12}$$

$$P(\text{lost}) = 1 - P(\text{win})$$

In this formula, f is the fraction of your wealth to wager on a bet. b is the amount of money you would receive on a win, not counting the wager itself. f can range from -1 to 1; where positive values mean a *back bet* should be placed; i.e. the bet should be made for the horse to win the race. Negative values for f means that a *lay bet* should be made. Most traditional bookmakers only let you make back bets but Betfair allows for lay bets, with the caveat that you do not get the same odds on lay and back bets except for BSP (and you do not know what BSP is until the race begins).

Kelly's work was seminal and strategies based on Kelly's criterion are now commonly used in finance. However, we need something more sophisticated for betting purposes than the simplistic formula 12 above. The reason is that the probabilities we use in the formula are *estimations* of probability and our models frequently get them wrong, compared to market odds b, which is another estimation of probability.

Therefore, we will not be using the simple Kelly formula above. We, in fact, tried using it, plugging estimations from our model described in previous section and using plain BSP price as b in the formula but results were disappointing; often ending up in less wealth than we started with. Instead, we will fit a

strategy model that uses Kelly criterion as a component but has the capability to take into account the error in our estimations.

Our betting system can be described with the following steps:

1. Goal: maximize wealth relative to starting wealth. For this scheme we set starting wealth at 1 unit of money.

2. Only consider a horse for betting if our model predicts higher probability of winning than implied by starting price.

3. Bet f^* fraction of wealth, where

$$f^* = \text{CAP}(\text{KellyCriterion}((\text{BSP})w_0 + w_1, P(\text{win})), w_2)$$

   w_0, w_1 and w_2 are optimizable parameters and optimized for maximum wealth at the end. $P(\text{win})$ is the estimated probability of winning as predicted by our model. $\text{CAP}(x, y) = x, \text{if } x < y,$ otherwise $\text{CAP}(x, y) = 0$.

The model on step 3 determines how much wealth we wager at each betting opportunity. The loss function on this model is non-smooth and non-differentiable. However, the number of parameters is very low (there are only 3 parameters) so it becomes feasible to use non-derivative based optimization. We used CMA-ES by N Hansen et al. [14], described as "derandomized evolution strategy with covariance matrix adaptation", which is a state-of-the-art algorithm for black-box optimization. CMA-ES can be seen as a type of genetic algorithm, with some useful heuristics to converge as quickly as possible.

The parameters on model 3 are designed to account for error in our model. w_2 puts a cap on how much wealth we are willing to wager per bet. w_0 and w_1 define a very simple linear regression on the input of Kelly criterion to adjust the amount to wager. We used softplus function ($\log(1 + e^x)$) on these parameters to force them to be positive (i.e. we actually optimize $\hat{w}_0$ and $\hat{w}_1$ where $w_0 = \log(1 + e^{\hat{w}_0})$ and $w_1 = \log(1 + e^{\hat{w}_1})$).

We found that this model is very prone to overfit; even with only three parameters the model was able to find rare cases where it would bet almost all wealth on very high odds and instantly increase our wealth by extreme amounts. The model would mostly not bet on other cases. Therefore we found it necessary to regularize the model in three ways: first, we used L2 regularization to prevent any of the parameters from being optimized to extremely high or low numbers; second, we heavily penalized models that bet less than 10% of time and third: we artificially reduced returns from successful bets that bet for odds that were higher than 10 (i.e. implied probability of winning by starting prices was less than 0.1). These measures seem to have been enough to make the model more well-behaved.

Finally, we considered the two types of bets: lay bets and back bets. Back bets bet for someone to win the race; you receive reward if your choice wins. Lay bets bet against; if your choice loses you get a reward. As we have already mentioned, both of these bet types are available on Betfair.

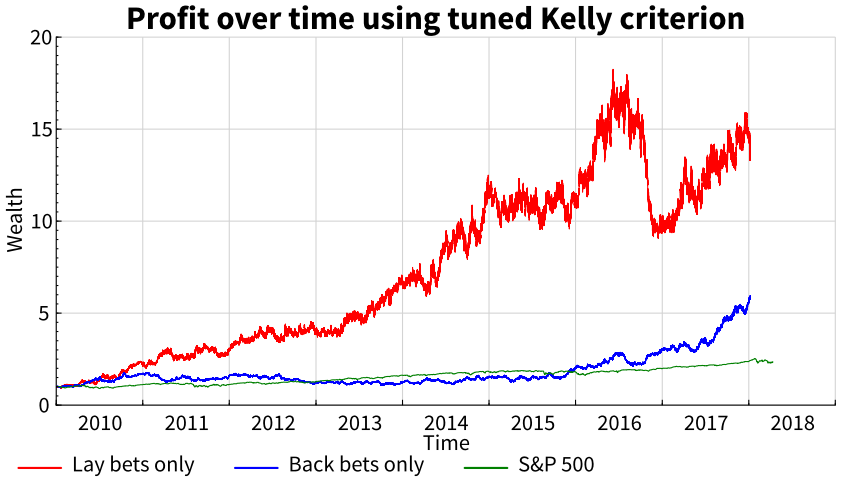

Figure 9: Betting returns from lay and back bets against BSP, compared with S&P 500 index. The returns on this graph account for 6.50% Betfair commission.

Figure 9 shows the evolution of wealth over time, using the tuned betting strategy with out-of-sample estimated predictions for each race and horse with the model we built in previous sections. This shows lay bets (the highest line, in red), back bets (blue line, for the most part on the middle) and for comparison, the Standard & Poor 500 index fund. All strategies start at 1 January 2010 at wealth 1.

First thing to note is that according to this model, our betting system is, on long-term, profitable, if done using lay bets. In fact, the lay bet system makes our wealth over 15 times higher than it was originally. The back betting, in the end, generates money but it has a years-long period of not significantly making returns between around 2012 and 2015. Both strategies are highly volatile. We noticed our model tends to have many more opportunities to do lay bets than back bets which may explain the volatility in lay betting.

For both back and lay betting there is a significant downward cliff at late 2016, where wealth almost halved for lay betting, but after that, returns are steady and positive. Some of the features we used in our betting model depend on time; for example, average horse position is more accurate for later years because there is more historical data to build this feature out of. This could, possibly, explain the returns for back betting during late years. The presence of this cliff is problematic if we wanted to empirically test our betting system: it suggests there may be time periods where months of betting results in significant loss, yet long-term the model would still be profitable.

There are many other caveats to this result. First of them is that the way we did out-of-sample estimations does not match up with reality. This is the same issue we had with our sparse model in section 5.2. We split the data randomly by

race; this means we may be training a model on future races and using past races to estimate out-of-sample performance. Alternative scheme, that would produce more accurate out-of-sample estimations would be to split the data into chunked time series, train a model up to date N_{train} and do out-of-sample estimation on dates $N_{\text{test}} > N_{\text{train}}$. We could not perform this scheme mostly because of time constraints; just building a single model end-to-end takes hours on our testing machine and the problem becomes worse with cross-validation that requires many folds. Moreover; many jockeys and horses compete during successive days or even on the same day and this happens often. To get accurate out-of-sample estimates that do not underestimate the performance of the betting strategy, we would have to train on very short time periods in our validation set and build hundreds of models, with each validation set consisting only 1 or 2 days worth of data in each fold.

A second caveat, although one that we account for, is that Betfair imposes a commission on bets. For bettors in Finland, at the time of writing of this, April 2018, the base commission, 6.50% on net winnings, with a discount scheme that can waive some of the commission. The profits on figure 9 assume a 6.50% commission on all net winnings. Betfair has complicated rules for commission on highly profitable customers so the commission is not entirely predictable.

A third caveat is that this model technically uses a feature that is only available after the race has started: the BSP price. We assume that the model could replace the starting price with whatever the posted market odds are at the time of betting and still improve on them. We could also wait until the last minute to bet; we saw in section 3.7.3 where we investigated market odds before the race begins that the market is almost as smart as starting prices just before the race. On the other hand, betting as early as possible may be beneficial; we would bet against a less sophisticated market.

Fourth caveat is that betting on the market may in itself affect the odds. The effect is likely small when the wagers are small but this can definitely change the odds if you start betting using thousands of dollars. In comparison, you likely can invest in S&P 500 index fund with millions and not likely affect the price of the index.

Alternative betting strategies might change results or make them more reliable; for example, the proportional betting strategy as described by Cover and Thomas [11] might result in more reliable returns.

Can you build a profitable betting system based on historical data? The answer: *probably yes; our data shows significant returns but those results come with many caveats attached.* Benter described the data-driven betting in 2008 [4] as the "golden age of computerized betting", with large returns being possible at the time. It looks like this may still be true today but to a much lesser extent. We believe that, given the complicated system and possibility of mistakes, a long-running empirical testing of the betting system is required to draw more confident conclusions that the system truly works in the real world.

7 Conclusions

We will conclude and summarize the most important results in this research in this section, along with ideas for future work.

Overall, we find that it is quite difficult to build a predictive system for horse racing that is predictive enough to beat the market odds on a betting market. There is quite a lot of good data available and you can make a relatively good predictor of results but this predictor is no good for betting purposes as-is.

Some of the "common wisdom" factors commonly used for betting are not entirely justified. Draw bias and course features definitely have effect on some race courses but the effect is small on majority of race courses, with only some race courses having significant effect. The pedigree information is even less useful: while there is correlation between the performance of a parent and their descendants, it ends up not being quite so useful for betting, other factors are more important.

For the purpose of predicting horse race results, the most useful information can be found by looking at the past performance of a horse. A simple normalized average horse position in past races by itself is much more informative than other observable properties of a horse such as its age or how much weight they are carrying in a race. Even if the horse is racing for the first time, jockey and trainer past performance are also very informative compared to other features you can look at.

A sparse model can be built that has one optimizable feature for each horse, jockey and trainer. One advantage of this is that it allows building a model that ranks all horses in the data set from worst to best directly; this is useful to compare two different horses in a race or otherwise to test correlations between some feature and the ranking of a horse overall. It is quite difficult to fight overfitting with a sparse model like this; most horses do not have many races behind them and estimations are bad when we have small sample sizes for each horse. Fortunately, despite the overfitting problem, our out-of-sample estimations for a sparse model still beat most other engineered features.

Even when we combine the best features and our sparse model together, we do not end up with a model that can beat market odds; in fact, combined sparse and dense model is worse than just using a dense model (the sparse model would need more sophisticated regularization mechanisms to fight overfitting). However, what we *can* do is stack our predictions on top of betting market starting prices, in order to improve the predictions of the betting public. The edge over the market we can attain this way is very small and the features that are useful for stacking are quite different from features that are useful for predicting horse race results.

The stacked model attains positive returns in our out-of-sample estimations using a strategy that only does lay bets. The stacked model is used with a betting strategy that aims to choose optimal size of bet based on how far off our prediction of the probability of winning is off with the existing market odds and caps on how much it will be willing to bet. This betting strategy itself is a model that is optimized, using Kelly criterion as a component. The out-of-

sample estimations, unfortunately, have many caveats attached to them that may make our estimations overly optimistic about returns. An empirical test of the system would be useful but it likely requires months of betting before there is enough data to draw conclusions. This is because our edge over the market is very small.

One limitation of this thesis is that we found ourselves unable to do more robust out-of-sample estimations of our betting system and our predictive model. We split our training data into training set and validation set by splitting them randomly by race-level. A better idea would be to split the data into training and validation set by time; train with data up to date N and validate on date N+1. Increase N by one and repeat until all data has been validated. The reason we did not do this is that in most cases time to exhaustively estimate out-of-sample results for all the data day-by-day would take too long with our codebase. A future work may want to pay attention to make sure that the training procedure is computationally *efficient.*

Another limitation is in our betting strategy. We used a relatively simple betting model that is designed to be resilient to errors in our horse race prediction model, to make sure the strategy won't ever decide to wager all the wealth. Kelly's work on proportional betting may lend itself to more theoretically clean ways to place wagers in the presence of modeling errors in the underlying horse race predictor and this could be an avenue for future work.

Finally, our betting winning computations are based off starting prices and our predictive model assumes it can use starting prices in its machinery that produces probabilities that each horse will win. We surmise that we can swap the starting price to some other price that is in place before the race starts and get similar results but we did not measure this for the lack of suitable data. Future work could collect data on betting prices *before* the race starts and investigate in what ways it behaves differently from starting prices, possibly opening up more profitable schemes.

Is it possible bet profitably using data you download off the Internet? The answer seems to be *yes* but it may not be worth the time and effort. An access to higher quality data could significantly improve our predictive model. The betting public is clearly better informed than the data sets we had for this thesis, as the market odds are much better predictors of horse race results than even our best model using dozens of diverse features. Only with a stacking model can we get an edge over the market and even that edge is very small.

```
horse=> SELECT advanced_going,COUNT(*) AS cnt FROM daily_races
        GROUP BY advanced_going
        ORDER BY cnt DESC LIMIT 15;

       advanced_going       |  cnt
----------------------------+-------
Standard                    | 24826
Good                        | 13937
Good to firm                | 10713
Good-good to firm in places |  8370
Good to firm-good in places |  6854
Soft                        |  6038
Good-good to soft in places |  4559
Good to soft                |  4183
Heavy                       |  3636
Good to soft-good in places |  3541
Good to soft-soft in places |  3260
Soft-good to soft in places |  2950
                            |  2940
Soft-heavy in places        |  2245
Good to firm-firm in places |  1639
(15 rows)
```

Table 14: Going values in Betwise data set

A Normalization of "going" values in Betwise data set

The way "going" is represented in Betwise data set is inconvenient and not easily machine-understandable. The problem is that the going is represented as a human-readable description. Table 14 shows counts for top 15 "going" values in the data set:

These are only for the table *daily_races*. Betwise also has a table for historic races, where the going is much more easily readable; there are only 17 possible values for "going" and only 13 for UK and Irish races. However, we cannot freely use historic races data because that would make a betting system using the data nonviable; the historic data is only available *after* the race is over. We can, however, join daily races with historic races and map most human-readable descriptions to the value system used in historic tables and then manually assign a value for them. Most future races still use at least one of the existing human-readable descriptions in *daily_races* table so we can assign feature values properly even for races for which we do not have historical data.

The 13 "going" entries, the absolute count in number of races with that going, and the values that the features take in our feature analysis (see section

Going	Count	Value
Firm	9837	1
Good to Firm	205023	0.9
Good	302528	0.8
Good - Yielding	8817	0.7
Good to Soft	143279	0.6
Standard	241134	0.5
Yielding	17686	0.5
Yielding - Soft	8210	0.4
Soft	136835	0.3
Soft - Heavy	12103	0.2
Slow	382	0.1
Heavy	56983	0
empty	449	0.5

Table 15: Different going values and numerical value assigned to them for feature analysis purposes.

4.1) are listed in table 15. These values come from our intuition; they are not based on any formal analysis. We believe that as long as they are vaguely correctly ranked, our analysis on them should not be significantly impaired.

The values listed on table 15 are used as a variable to represent a going value in all our analysis that uses the going.

B Course effects by race course

The table 16 in this appendix lists all race courses in the order of how much the inherent properties of the course influence results. It answers the question: how much do the properties and starting conditions of a horse race course affect the horse race results, disregarding inherent properties of the horses themselves? For example, if the randomly assigned stall numbers on the course are not "fair" (i.e. the course exhibits *draw bias*), the score on this list would be high. Only top 16 are listed; for the rest of the courses the different in goodness-of-fit starts to get meaningless.

Course name	P-L horse feature only	P-L horse and course features	Difference %
Sligo	0.022	0.040	79%
Chester	0.018	0.026	42%
Ballinrobe	0.038	0.048	41%
Wexford	0.039	0.054	36%
Wetherby	0.033	0.046	35%
Listowel	0.029	0.038	33%
Down Royal	0.028	0.037	31%
Pontefract	0.019	0.025	31%
Galway	0.029	0.037	30%
Clonmel	0.049	0.063	28%
Limerick	0.036	0.045	24%
Ascot	0.011	0.013	23%
Roscommon	0.032	0.039	22%
Cork	0.023	0.027	22%
Fairyhouse	0.027	0.033	22%
Espom Downs	0.013	0.016	21%

Table 16: Goodness-of-fits for different race courses. "P-L horse feature only" refers to goodness-of-fit of a model trained only with the average past horse position. "P-L horse and course features" includes course-specific features. "Difference" is the relative difference between the goodness-of-fit values and the table is sorted according to this last column (with strongest course effects at the top of the table). 100% would mean the goodness-of-fit using course features improves goodness-of-fit to twice the value of a model that only considers average past horse position. 0% implies the course features are useless.

References

[1] Mukhtar M Ali. Some evidence of the efficiency of a speculative market. *Econometrica (pre-1986)*, 47(2):387, 1979.

[2] Mukhtar M Ali. Probability models on horse-race outcomes. *Journal of Applied Statistics*, 25(2):221–229, 1998.

[3] RE Barlow and HD Brunk. The isotonic regression problem and its dual. *Journal of the American Statistical Association*, 67(337):140–147, 1972.

[4] William Benter. Computer based horse race handicapping and wagering systems: A report. Technical report, World Scientific Publishing Co. Pte. Ltd., 2008.

[5] MM Binns, DA Boehler, and DH Lambert. Identification of the myostatin locus (mstn) as having a major effect on optimum racing distance in the thoroughbred horse in the usa. *Animal genetics*, 41(s2):154–158, 2010.

[6] Ruth N Bolton and Randall G Chapman. Searching for positive returns at the track: A multinomial logit model for handicapping horse races. *Management Science*, 32(8):1040–1060, 1986.

[7] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[8] Alasdair Brown. Evidence of in-play insider trading on a uk betting exchange. *Applied Economics*, 44(9):1169–1175, 2012.

[9] Randall G Chapman. Still searching for positive returns at the track: Empirical results from 2,000 hong kong races. In *Efficiency of racetrack betting markets*, pages 173–181. World Scientific, 2008.

[10] Randall G Chapman and Richard Staelin. Exploiting rank ordered choice set data within the stochastic utility model. *Journal of marketing research*, pages 288–301, 1982.

[11] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[12] Karen Croxson and J James Reade. Information and efficiency: Goal arrival in soccer betting. *The Economic Journal*, 124(575):62–91, 2014.

[13] Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

[14] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

[15] Donald B Hausch, William T Ziemba, and Mark Rubinstein. Efficiency of the market for racetrack betting. In *THE KELLY CAPITAL GROWTH INVESTMENT CRITERION: THEORY and PRACTICE*, pages 663–680. World Scientific, 1980.

[16] J. L. Kelly. A new interpretation of information rate. *Bell System Technical Journal*, 35(4):917–926, 1956.

[17] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February 1966.

[20] Duncan R Luce. Individual choice behavior. 1959.

[21] Nick Mordin. *Winning Without Thinking: A Guide to Horse Racing Betting Systems*. Aesculus Press Ltd, 2002.

[22] Robin L Plackett. The analysis of permutations. *Applied Statistics*, pages 193–202, 1975.

[23] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

[24] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.