Prediction of Swedish Harness Racing
A Bachelor Thesis in Mathematical Statistics

Jonas Josefsson and Martin Hellander
Vehicle Engineering, KTH

May 21, 2013

Abstract

Harness racing is a sport where betting most often is done based on historical performances and known conditions from each horse. With up to 12-15 horses in each race and with a quite large set of data for each horse, harness racing seemed to be very suitable for some statistical modeling and regression analysis. The main goal of this project was to construct a model that predicts the outcome of a race better than the odds. To achieve this, many different covariates, and combinations of them, have been tested. Also different types of regression methods, such as logistic regression have been tested in order to find the best model. A big challenge has been to collect a very large amount of useful data and handling it in an efficient way. In the end several models had been developed whereof the best ones made better predictions than the odds. Also a few betting strategies have been developed in order to investigate the possibility of making money by using the models. At least one of them seem to provide a good return.

Acknowledgements

We would like to thank Jonas Hallgren for guidance and support during this project.

Contents

Chapter 1

Introduction

1.1 Project

The objective of this project is to be able to predict the outcome of future races, especially focusing on betting in a single race and the game form of V75, where the goal is to predict the winners for seven races. To do this some mathematical models using multiple linear regression, as well as logistic regression, on historical data are to be developed. All with one mutual goal, to make better predictions than the odds. Since the model will be predicting the outcome of future races it is of great interest to know how to use the model for betting. Hence, a brief introduction to betting will also be included.

1.2 Swedish Harness Racing

In Sweden harness racing is a very popular sport where several kinds of betting are possible. It is possible to bet on a horse and if the horse wins the race a return will be given to the player. The most popular game form, V75, there are seven races with about 10 to 15 horses in each race and the objective is, of course, to tell all of the winners. So with, for example, 12 horses in every race there are approximately 36 million different combinations of possible winners. Usually V75 has a turnover of 80-90 million SEK during every weekend[6]. That kind of money and, what it seems, the absence of useful mathematical models for predicting the outcome of a race makes it an interesting area to investigate.

1.2.1 The Betting System for V75

In each race you can choose a number of horses to place as winner. By including more potential winners the chance of winning will of course increase but, unfortunately, so does also the cost. When calculating how much a system will cost the number of rows are calculated and multiplied by 0.50 SEK. The number of rows are calculated as, $\#rows = \prod_{j=1}^{7} \#horses_j$, i.e. the product of the number of horses in each race. An example of a system which contains $3 \cdot 2 \cdot 1 \cdot \ldots \cdot 1 = 6$ number of rows and thus costs three SEK is shown in figure 1.2.1 below.

Figure 1.2.1: A V75 system[6].

In general, payouts are made for systems containing at least five winners. This means that a system that manages to find all seven winners often get multiple payouts for also containing several combinations of five and six winners. If the return for players with seven winners is too small systems containing five or six winners will not get any return.

1.2.2 The Betting System for Betting on a Single Horse

When betting on a single horse a return will be given if that horse wins its race. How big the return will be depends on the size of the bet as well as the odds of the horse. A high odds will give a big return while a small odds will give a small return. This is a very risky way of playing since it is only possible to win big if the bet is big.

1.3 Nomenclature

Below follows some explanations of commonly used harness racing terms.

Trot (trav)

A two-beat movement style of a horse where its diagonal pairs of legs move forward at the same time.

Car start (autostart)

All of the horses start behind a starting vehicle.

Volt start

The horses trot in circles in a specific pattern to hit the starting line as two groups, with the best horses in the rear group.

Gallop (galopp)

A prohibited movement style in Swedish harness racing. Might lead to disqualification[1].

Odds

A measure of how likely a horse is to win the race according to the players. A low odds meaning that the horse is more likely a winner.

Mare (sto)

A female horse.

Stallion (hingst)

A male horse.

Gelding (valack)

A castrate male horse.

Starting Points (startpoäng)

Points showing how well a horse has performed during the last 5 races. The points depend both on placement and earned money.

Place Percentage (platsprocent)

Showing how often the horse finishes at third place or better.

Chapter 2

Theory

2.1 The Linear Regression Model

A commonly used method for predictions based on historical data is the multiple linear regression model which is defined as

$$y_i = \sum_{j=0}^{k} x_{i,j}\beta_j + e_i, \ i = 1, \ldots, n \tag{2.1}$$

where y_i is an observation of the dependent random variable Y. The expected value of Y depends on the covariates x_j. The model parameters β_j is estimated from the data and e_i is the disturbance term. The first covariate, $x_{i,0}$, is often determined to be equal to one which creates a constant in the model, such as

$$y_i = \beta_0 + x_{i,1}\beta_1 + \ldots + x_{i,k}\beta_k + e_i, \quad i = 1, \ldots, n. \tag{2.2}$$

The linear regression model is often written with matrix notation, i.e.

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{e}. \tag{2.3}$$

2.1.1 Estimation

The estimated values of β are presented as $\hat{\beta}$. They are estimated by the OLS estimate (Ordinary Least Squares), which is proven to be the best linear unbiased estimator (the BLUE), i.e. it has smaller variance than all other linear estimators [2]. Since the estimator is unbiased it follows that

$$E\left[\hat{\beta} \mid \mathbf{X}\right] = \beta, \tag{2.4}$$

i.e., the expected value of $\hat{\beta}$ is equal to β. The definition of the OLS estimate is that it estimates the value $\hat{\beta}$ of β that minimises residual sum of squares $\hat{\mathbf{e}}^t\hat{\mathbf{e}} = |\hat{\mathbf{e}}|^2$. This is done by solving the so called normal equations $\mathbf{X^t}\hat{\mathbf{e}}$=0 for $\hat{\beta}$. It then follows that the OLS estimate of β is

$$\hat{\beta} = \left(\mathbf{X^t X}\right)^{-1} \mathbf{X^t Y}. \tag{2.5}$$

Where an unbiased estimator of the regression variance

$$\hat{\sigma^2} = s^2 = \frac{1}{n-k-1} |\hat{\mathbf{e}}|^2 \tag{2.6}$$

where n is the number of observational data and k the number of covariates. So

$$E\left[s^2 | \mathbf{X}\right] = \sigma^2 \tag{2.7}$$

is satisfied for the estimated variance as well[3].

These estimations can be done easily in MATLAB with the *regress* function. These functions also report statistics for the regression, such as the sum of squared errors, estimated standard errors for each β etc.

2.1.2 Standard Error of Beta

The covariance matrix for $\hat{\beta}$ is defined as

$$Cov\left(\hat{\beta} \mid \mathbf{X}\right) = \left(\mathbf{X^t X}\right)^{-1} \sigma^2 \tag{2.8}$$

where σ^2 is the variance which can be estimated with s^2 according to chapter 2.1.1. Hence, the covariance matrix is estimated as

$$\hat{Cov}\left(\hat{\beta} \mid \mathbf{X}\right) = \left(\mathbf{X^t X}\right)^{-1} s^2. \tag{2.9}$$

The estimated standard deviation, i.e. the standard error of a parameter$\hat{\beta}_j$, is then$SE\left(\hat{\beta}_j\right) = \lambda_j s$ where λ_j^2 is the j:th diagonal element of the matrix$(\mathbf{X^t X})^{-1}$.

2.1.3 Prediction

When the values of β have been estimated the model can be used for prediction, i.e.

$$y_p = \mathbf{x_0}\hat{\beta}, \tag{2.10}$$

where y_p is the predicted value and $\mathbf{x_0}$ a row matrix containing the known covariates. When predicting is the purpose of a model things like multicollinearity or endogeneity does not have to be considered. Theses things only have to be considered if the purpose is to investigate how a covariate x influence a dependent variable y which is not the case in this project[3]. Therefore things such as confidence intervals or hypothesis testing for parameters will not be investigated.

2.2 The Logistic Regression Model

A different type of regression is the logistic regression, also called the logit. The logit can be used in cases where the dependent variable, y, is naturally a probability. The logit is defined as

$$y_i = \frac{\exp(x_i\beta)}{1+\exp(x_i\beta)} = p(x_i\beta), \tag{2.11}$$

where y_i is given by dummy variables such that it is equal to one if the event occurred and zero otherwise. The estimation of β is estimated by the Maximum Likelihood estimation. A logistic regression can be done in MATLAB using the *glmfit* function.

Chapter 3

Data

3.1 Collecting and Sorting the Data

The data used in this project was collected from `travsport.se`, which is the official web page of Swedish harness racing. It contains information and historical data of every Swedish registered horse, driver, trainer etc. A MATLAB script was written to obtain the data needed in the analysis. The script downloaded the HTML code for all pages containing necessary information and saved it to text files. To extract the important data from the HTML code another MATLAB script was written. About 2600 of the horses and their data, such as results, times, earnings, etc. were picked out to be part of a 1x2600 struct array with one field for each type of data. This struct was saved and later used for running regressions and predicting results.

3.1.1 Data Used when Running the Regression

When running regressions to obtain the estimated beta values a structure similar to a large excel sheet was desired, i.e., one column for each covariate and one column for the dependent variable. To get the data in this form a new struct, containing all covariates as if they were from a single horse, was created. Since the regression must not contain the more recent races, which were to be used for testing the model, all data from a certain date to the present were removed. Besides structuring the data as desired, a lot of time was spent on creating new covariates to give more options when modelling.

For two of the models the data had to be sorted according to race date and race number to be able to compare horses within each race. In those cases the amount of data was reduced so that only the races with data for at least eight of the horses, the winner being one of them, were included.

3.1.2 Data Used when Running the Prediction

To run the prediction a structure as the one used when running the regression was desired. So, as in that case, a new struct was created. Though this time the struct did not contain anything except the horses in the race and the data belonging to the day of the race which were to be predicted, i.e. one row per horse.

3.1.3 Problems with the Data

During this project several unknown obstacles concerning the data have been encountered resulting in replanning of the project. For example when horses have been competing abroad the data looks different and some data, such as starting numbers, are missing. Another problem is that data on foreign horses are only available for a few days before the race to a few days after the race. This fact heavily reduces the number of races were all horses are available. The biggest problem was that during the project `travsport.se` added a block making it impossible to enter the web page as often as needed. This meant that the MATLAB script for downloading the data did not work nearly as quick as before making it impossible to access the data in a reasonably short time. At the beginning of the project, when this block was not in use, enormous amounts of data could be collected in a really short time. But when more data was needed this was, and still is, a very big problem which in the end stopped us from collecting the preferred data.

Chapter 4

Modeling

4.1 Modeling

To be able to compare models and evaluate them the regressions are analysed and predictions are made on historical observations, which are not part of the regression. When making predictions it is important to be cautious since it is very easy to include something in the model that should not be there. Perhaps, as in this case, historical observations are used to be able to compare the predicted values with the true values, then future observations must not be included in the regression and the used covariates must not be unknown before the actual observation took place. If you include these, forbidden, kind of things it will be like ”cheating”, i.e. predicting a future outcome based on what is supposed to be unknown future data.

4.1.1 Data

In the beginning of this project a regression was run for one race at a time. This was done by regressing data only containing horses from each race separately and then predicting the outcome of the race. But when only using data from 12-15 horses which gives around 100 observations the beta estimations are not very reliable. So in order to get better estimations the amount of data was increased from only containing horses from each race to containing horses from each race day. In a race day there are about 85 horses competing which gives a couple of thousands observations. After some performance issues the number of observations was increased once again reaching almost 60 000.

4.1.2 Evaluation of the Models

By evaluating the models it is possible to find out which one is the best. To find all covariates which affect the dependent variable many covariates must be tested. To decide whether a covariate should enter a model or not the AIC value was computed. AIC, Akaike information criterion, is a measure of the goodness of fit of the model. Hence, the model with the best AIC value is the preferred model. AIC is defined as

$$AIC = 2k - 2ln(L) \tag{4.1}$$

where k is the number of parameters in the model and L the maximised value of the likelihood function for the model. The best model according to AIC is the model with the lowest AIC value. AIC grows with a larger number of parameters, therefore it is not just a measure of the goodness of fit but also helps in preventing overfitting[8]. Overfitting means that the model fits the data so well that it also describes the random errors. Having an overfitted model which describes random errors of a data set is of course not desired when using the model in a predictive purpose since this will most likely worsening the result of the prediction. However even a model which is not overfitted and has a good fit of the data does not necessarily need to be a good model for predicting.

The accuracy of models using logistic regression can be evaluated with ROC (receiver operating characteristic) curves. A ROC curve is a plot with x-axis defined as, in our case, $P(horse\ predicted\ as\ winner\ does\ not\ win)$ and the y-axis is defined as $P(horse\ predicted\ as\ winner\ wins)$. The accuracy of the model is then calculated as the area under the curve. This means that if the curve is a straight line from the origin to $(x, y) = (1, 1)$ the accuracy of the model is 50 %, i.e. guessing the winner will give a result as good as the result predicted by the model. This makes the model useless which is, of course, not a desired result. On the other hand, if the model is perfect, i.e., 100% accurate, the plot will go in a straight line from the origin to $(0, 1)$ and then in another straight line to $(1, 1)$. Hence, the objective is to construct a model whose ROC curve has an area under which is as close to 1 as possible[9].

4.2 Covariates

Below is a list of all the covariates which either come from the data or has been calculated based on the data.

4.2.1 Taken From the Data

Odds

The odds according to the players.

Car start

Dummy with ones where the race had car start and of course zeros where the race had volt start.

Mare

Dummy variable indicating that the horse is a mare.

Stallion

Dummy variable indicating that the horse is a stallion.

Distance

In a volt start i.e. not behind a vehicle the horses are starting in two or three groups behind each other. This means that some of the horses are running further than the horses in the first starting group. *Distance* is a dummy variable telling if the horse will run further than some of the other.

4.2.2 Calculated From the Data

Age

The age of the horse.

Win percentage

Lifetime win percentage at current date. It is defined as $Win\,percentage_i = \frac{1}{n-j}\sum_{j=i+1}^{n} won\,race_j$ where $won\,race_j$ is equal to one if the horse won race number j.

Place percentage

Lifetime place percentage at current date. It is defined as $Placepercentage_i = \frac{1}{n-j}\sum_{j=i+1}^{n} placerace_j$ where $placerace_j$ is equal to one if the horse was placed better than fourth in race number j.

Starting points

A value based on the latest five performances including both placement and earned money. Used by ATG, which is the company that provides the betting.

Money

The covariate $Money$ is the average money earned per race. It is defined as $Money_i = \frac{1}{n-j}\sum_{j=i+1}^{n} money_j$ where n is the number of races in a horses carrier. A high value of $Money$ should indicate that a horse have performed well in races with high prize money.

Time

The covariate $Time$ is the best historical time for a horse on the current distance.

Win Shape

The covariate $Win\ shape$ is the average placement based on the three latest results. It is defined as $Win\ shape_i = \frac{1}{3}\sum_{j=i+1}^{i+2} placement_j$ i.e. a low $Win\ shape$ should indicate that a horse is in a good shape based on the result.

Money Shape

The covariate $Money\ shape$ is the amount of money earned in the three latest races. It is defined as $Money\ shape_i = \frac{1}{1000}\sum_{j=i+1}^{i+2} money_j$ i.e. a high $Money\ shape$ should indicate that a horse is in a good shape based on its recent earnings.

4.2.3 Made-up and Calculated From the Data

Very good, good and decent driver

Dummy variables indicating how good the driver is. This is based on a driver ranking which is based on the drivers historical win percentage. Specific percentage levels indicate if it is a very good, good or decent driver.

Very good, good and decent horse

Dummy variables indicating how good the horse is. This is based on a horses historical win and place percentage. Specific percentage levels indicate if it is a very good, good or decent horse.

Starting number rank

The historic win percentage for each starting number and method. This is not linear, e.g. starting number three is both better than starting number one and five.

Distance win fit

A dummy variable indicating that the horse performs well at the current distance based on the win percentage.

Distance place fit

A dummy variable indicating that the horse performs well at the current distance based on the place percentage.

Season win fit

A dummy variable indicating that the horse performs well at the current season based on the win percentage.

Season place fit

A dummy variable indicating that the horse performs well at the current season based on the place percentage.

Starting method win fit

A dummy variable indicating that the horse performs well with the current starting method based on the win percentage.

Starting method place fit

A dummy variable indicating that the horse performs well on the current distance based on the place percentage.

4.3 The Models

Three models were developed with three different approaches. Two multiple linear regression models with different kind of data sets and one logistic regression model. In the first two models the placement of the horse is the dependent variable and in the third the dependent variable is the probability to win the race. Model 1 deserves a chance but more effort was put into developing model 2 and model 3 since the possibility of making good predictions were considered better for them.

4.3.1 The First Model

The first model is a multiple linear regression model estimated with data from many horses including almost 100 000 observations from randomly chosen horses.

Model Selection

As mentioned in section 4.1.2, AIC is a good measurement when deciding which covariates to include in a model. For a linear regression model the AIC value is calculated as

$$AIC = 2k - 2ln\left(L\left(\beta, \sigma \mid X, Y\right)\right).$$

So with the assumption that $e_1, \ldots, e_n \sim N\left(0, \sigma^2\right)$ the Likelihood function is

$$ln\left(L\left(\beta, \sigma \mid X, Y\right)\right) = -\frac{n}{2}\left(ln\left(2\pi\right) + ln\left(\sigma^2\right)\right) - \frac{1}{2\sigma^2}\left\|Y - X\beta\right\|.$$

With $\beta = \hat{\beta}$ the likelihood function is maximised since the OLS estimator is the BLUE, see chapter 2.1.1. And with the MLE estimate of the variance

$\hat{\sigma^2} = \frac{SSE}{n} = \frac{\|Y-X\beta\|}{n}$ it follows that

$$AIC = 2k + n\left(ln\left(2\pi\right) + 2ln\left(\hat{\sigma}\right)\right) + n. \tag{4.2}$$

From Table 4.1 it can be seen that when a covariate, in this caseAge, increases the AIC value it is not included in the model. This is done in an iterating process until all of the covariates have been tested. The procedure continues until there are no covariates left which decreases the AIC value. The root mean square of the residual i.e.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2} = \sqrt{\frac{SSE}{n}} \tag{4.3}$$

is also presented in Table 4.1. This method is not perfect since there are many untested combinations of variables which might give a better result.

Covariates	$\mathbf{AIC_{REG}}$	$\mathbf{RMSE_{REG}}$
$Odds$	149582.3	0.800
$Odds + Win\,shape$	149317.8	0.799
$Odds + Win\,shape + Age$	149319.8	0.799
$Odds + Win\,shape + Win\,percentage$	149150.2	0.797
$\vdots$	$\vdots$	$\vdots$
$Odds + Win\,shape + ... + Start\,number\,rank$	148737.6	0.795

Table 4.1: Model 1 selection. The AIC value for the model and the RMSE for the data is presented for different models.

Model one

The resulting model is as follows

$$\begin{aligned} \ln\left(placement_i\right) &= \beta_0 + Odds \cdot \beta_1 + Win\,shape_i \cdot \beta_2 + Win\,precentage_i \cdot \beta_3 \\ &+ \quad Place\,percentage_i \cdot \beta_4 + Very\,good\,driver_i \cdot \beta_5 \\ &+ \quad Good\,driver_i \cdot \beta_6 + Decent\,driver_i \cdot \beta_7 + Start\,number \\ &\quad\; rank_i \cdot \beta_8 + e_i, \qquad i = 1, \ldots, n \end{aligned}$$

So the placement is predicted as

$$placement_i = \exp\left(\hat{\beta}_0 + ... + Start\,number\,rank_i \cdot \hat{\beta}_8\right), \quad i = 1, .., n. \tag{4.4}$$

4.3.2 The Second Model

The second model is estimated with another set of data. This set contains whole, or almost whole, races instead of random observations in different races. By using this data set a comparison between the horses within a race is possible. For example a horse's average earning can be compared to the other horse's average earnings within the race. If this wouldn't be compared within the race it wouldn't make a difference because in a race with bad horses a horse with a low average earning will win and in a race with good horses a horse with a large average earning will win. This is the reason why more effort was put into developing model 2 than what was put into developing model 1.

Model Selection

To increase the possibility of finding a good model the covariates are not only used alone, they are also combined as multiplications between them. The squares and the square roots of the covariates were also tested. Since this made the number of possible covariates very large some kind of method to determine which covariates to include in the model had to be used. A very intuitive way to do this was to begin with an empty model and add one covariate and calculate the AIC value. After doing this the covariate was replaced by another covariate and the new AIC value was calculated. By repeating this for all covariates it was possible to find the covariate which influenced the model most. The best covariate was then included in the model and by repeating this procedure the model grew by one covariate, i.e. the best one, at a time. Although this method was not perfect since the covariates may affect each other differently when used in different combinations this was considered as a sufficiently good method and was therefore used when creating the model. The development of the model is presented in Table 4.2 together with the corresponding AIC value. The root mean square of the residual, equation 4.2, is also presented in Table 4.2.

Num. cov.	AIC_{REG}	$RMSE_{REG}$
1	19575.9	0.7354
2	19360.4	0.7263
3	19246.4	0.7216
4	191767	0.7186
⋮	⋮	⋮
138	18892.3	0.7031

Table 4.2: Model 2 selection. The AIC value for the model and the RMSE for the data is presented for some different number of covariates.

As mentioned earlier AIC has a penalty for so called overfitting or over-parameterization. This can be seen in figure 4.3.1. A suitable number of covariates appears to be somewhere between 20 and 60. So models with at most 45 covariates were considered not overparameterized during the rest of the project.

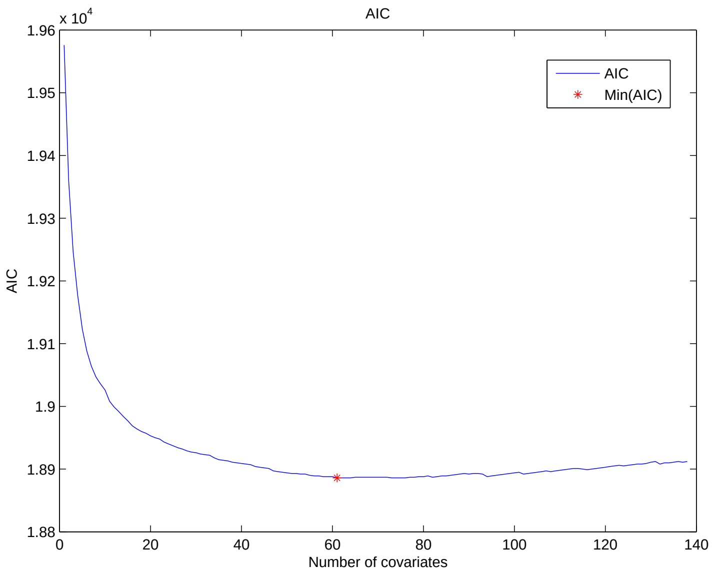

Figure 4.3.1: Plot of AIC.

Model two

Following the model selection method the 35 best covariates, according to AIC, was selected. By choosing 35 covariates instead of 45, for which the

AIC value was just slightly lower, the risk of overfitting decreased. This gave the model

$$\ln(placement_i) = \beta_0 + Odds_i \cdot \beta_1 + \ldots + Time_i \cdot \beta_k + e_i, \quad i = 1, \ldots, n. \tag{4.5}$$

The placement can now be predicted as

$$placement_i = \exp\left(\hat{\beta}_0 + \ldots + Time_i \cdot \hat{\beta}_k\right), \quad i = 1, .., n. \tag{4.6}$$

4.3.3 The Third Model

The third model was estimated with the same special data set as the second model. Though, since this model uses logistic regression the dependent variable, y, must have a binary appearance. This meant that instead of containing the results, as in the two previous models, y was now true or false. True meaning that the result was equal to one, i.e. the horse won the race, and false for any other results. Hence Y was now a vector with ones where the result was equal to one and zero for all other results.

Model Selection

As mentioned in section 4.2.1, when evaluating the accuracy of models based on logistic regression the area under the ROC curve, AUC, is a good measure. Since the curves were very much alike it was difficult to distinguish them from each other by just looking at the plots. Instead, the area under the curve was calculated for each model and then compared. The covariates were chosen in the same iterative process as for model two. Though in this case AUC was supposed to be maximised in contrast to the development of model two where AIC was minimised. The result of the model selection is presented in Table 4.4 together with its corresponding AUC value and RMSE.

Num. cov.	$\mathbf{AUC_{REG}}$	$\mathbf{RMSE_{REG}}$
1	0.7861	0.2921
2	0.7864	0.2920
3	0.7893	0.2917
4	0.7888	0.2917
$\vdots$	$\vdots$	$\vdots$
45	0.7855	0.2902

Table 4.3: Model 3 selection.The AUC value for the model and the RMSE for the data is presented for different number of covariates.

In figure 4.3.2 is the ROC curve of the model presented.

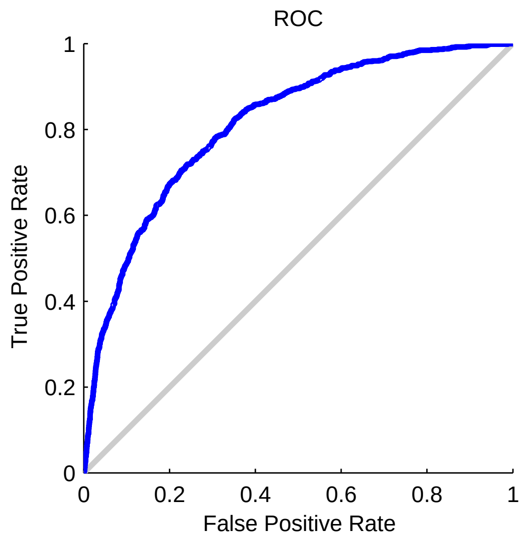

Figure 4.3.2: Plot of ROC.

Model three

$$y_i = \frac{\exp\left(\beta_0 + ... + Time_i \cdot \beta_k\right)}{1 + \exp\left(\beta_0 + ... + Time_i \cdot \beta_k\right)}, \qquad i = 1, \ldots, n \tag{4.7}$$

where y_i is equal to one if $horse_i$ won and equal to zero otherwise. So the probability that $horse_i$ should win is predicted as

$$p_i = \frac{\exp\left(\hat{\beta}_0 + ... + Time_i \cdot \hat{\beta}_k\right)}{1 + \exp\left(\hat{\beta}_0 + ... + Time_i \cdot \hat{\beta}_k\right)}, \qquad i = 1, .., n. \tag{4.8}$$

Chapter 5

Using the Model for Betting

Now when the final models had been chosen they can be used for betting. To do this it is important to decide whether the aim is to win often but not so big or more rarely but big. Or maybe just place a bet on a single horse in a certain race. In either case it is important to know which races that are tight and which races that are more easily predicted. This can be done in several ways. For example the probability of winning the race can be compared between the horses. The race can be considered as tight if some of the more probable winners have about the same probability to win. Also, if one of the horses has the probability of, say 50%, the race can be considered as more easily predicted. When knowing this it is possible to decide how many horses to include in every race to get the best chance of succeeding with the betting. Two strategies will be presented here, for more strategies see appendix A.

5.1 The Win Often Betting Strategy

If the the aim is to win often, i.e., to maximise the chance of predicting all seven winners, the problem can be considered an optimization problem. The equation we want to maximise is

$$\prod_{k=1}^{7} Pr(winner\ of\ race\ k\ included\ in\ the\ system) \tag{5.1}$$

under the condition that

$$cost \leq \prod_{k=1}^{7} \#horses_k \cdot 0.5 \tag{5.2}$$

which is the cost of the system, presented in section 1.3. This can be done quite easily in MATLAB either by using one of MATLAB's optimization functions or by doing it in a less efficient but more methodical way. The latter alternative was chosen since it makes it easier to modify the optimization progress to better fit our wishes. The first step in this more methodical way is to create an interval of the amount of SEK that will be used, say 400-500 SEK. Now MATLAB can create all systems with a cost included in the interval. Since the probabilities to win are known for all horses it is easy to create the probability to have the winning horse included in your system if you choose, say, three horses in a race. It is simply the sum of the probabilities of the three top ranked horses in that race. By using this, and all the game systems created by MATLAB, the solution which has the highest probability to include all seven horses at a cost of 400-500 SEK is calculated.

It is also possible to add more constraints such as

$$Pr(winner\ of\ race\ k\ included\ in\ the\ system) > p, \quad k = 1, ..., 7 \tag{5.3}$$

with p equal to any desired probability. This constraint is for high values of p sometimes impossible to satisfy due to the maximum amount of SEK available. But if it is possible to satisfy this constraint for, say, $p = 0.5$ this might indicate that there are a few very probable winners in each race. Hence, it might be a good idea to bet according to the system which satisfies this constraint since it will have a high probability to be correct for every race.

5.2 Betting in a Single Race

A third betting strategy that may be suitable is to bet on a single horse if this horse's probability to win according to the model is greater than its probability to win according to the odds. The probability to win for a horse according to the odds is

$$p_{odds} = \frac{1}{odds}. \tag{5.4}$$

However the sum of these probabilities in every race exceeds one, i.e. the probability that any of the horses will win is greater than one which is caused by the fact that the odds are lower than they should be. How much it exceeds one can be seen as the betting company's margin of safety which gives them an advantage against anyone who's betting. To be able to compare probabilities the normalized probability is calculated as

$$\tilde{p}_{odds_j} = \frac{p_{odds_j}}{\sum_{i=1}^{n} p_{odds_i}}, \tag{5.5}$$

where n is the number of horses in the race. Now the probabilities from the model can be compared to the normalized probabilities from the odds i.e.

$$p_j - \tilde{p}_{odds_j} \geq \lambda. \tag{5.6}$$

Which means that if this difference exceeds a limit λ horse number j has a favorable odds and may be worth betting on. Another way to look at it is calculating the expected values of how much you will win verses how much you will lose on a bet. With a bet of x SEK it follows that

$$E\left[money\ won\right] = p_j \cdot (odds_j - 1) \cdot x \tag{5.7}$$

$$E\left[money\ lost\right] = (1 - p_j) \cdot x. \tag{5.8}$$

And with the condition to win more than you lose it follows that

$$E\left[money\ won\right] > E\left[money\ lost\right] \iff p_j > \frac{1}{odds_j} = p_{odds_j}. \tag{5.9}$$

I.e. λ in equation 5.3 should at least be $\lambda = p_{odds_j} - \tilde{p}_{odds_j}$. Though this strategy has to be used with caution. If, for example, the difference in equation 5.3 clearly exceeds λ for some horse, but the probability that this horse would win is very low. Then it is easy to understand that even if this horse has really favorable odds, it is very unlikely that it will win and that betting would probably not result in a profit for a single bet. However, theoretically speaking, and with a model predicting better than the odds, when the number of bets b, like this, $b \to \infty$, you will make a profit. But if equation 5.3 is met combined with a criterion on the probability like

$$p_j \geq \mu \tag{5.10}$$

where μ is another limit, this strategy may be better.

Chapter 6

Results

6.1 Models

In table 6.1 model 2 and model 3 are compared based on their RMSE of the regression and the prediction data sets. Since the second model predicts placements these has to be converted to probabilities for comparison with model 3. This is done with MATLAB script[7]. To be able to compare these probabilities with the real result they all have to be normalized so that in each race the probability sum up to one, like $\tilde{p}_{odds_j}$ in equation 5.5. The RMSE is calculated as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} = \sqrt{\frac{SSE}{n}} \tag{6.1}$$

where y_i is equal to one if horse i won and zero otherwise and $\hat{y}_i$ is the normalized predicted probability.

Model	$\mathbf{RMSE_{REG}}$	$\mathbf{RMSE_{PRED}}$
Model 2	0.2840	0.2627
Model 3	0.2848	0.2644
Odds	0.2798	0.2977

Table 6.1: Models. The RMSE calculated on the data and on the prediction compared between Model 2, Model 3 and the Odds.

As the table shows both models predict better than the odds according to the RMSE of the prediction.

6.2 Predictions

Result of a predicted race by the second model can be seen in table 6.2. The first column is the real placements and the third is the predicted placements by the model two. These predictions are ranked, i.e. the horse with the lowest predicted placement is ranked as number one etc. and the same thing is done based on the odds.

Placement	**Ranked pred.**	**Pred. placement**	**Ranked odds**
1	2	3.6996	5
2	3	4.0609	6
3	5	4.4077	2
4	6	4.6083	3
5	4	4.2706	4
6	8	5.9351	7
7	10	6.2705	10
8	12	7.2113	12
9	9	5.9882	11
10	11	6.4413	9
11	7	5.0724	8
12	1	3.1044	1

Table 6.2: A predicted race. Placement is the actual placement of the horse. Pred. Placement is the placement predicted by Model 2 and Ranked odds is how the odds ranked the horses.

From this it can be seen that, in this case, the model performs a little bit better than the odds. The winner is ranked as number two by the model and number five by the odds. Both the model and the odds agrees on which horse that should win, however this horse performs really bad and ends up at last place.

The same race as in table 6.2 but predicted with model number three is shown below in table 6.3. So instead of placements the probability is predicted.

Placement	Ranked pred.	Pred. probability	Ranked odds
1	3	0.1243	5
2	7	0.0798	6
3	2	0.1567	2
4	3	0.1311	3
5	5	0.1177	4
6	6	0.0963	7
7	10	0.0127	10
8	12	0.0025	12
9	11	0.0092	11
10	9	0.0172	9
11	8	0.0732	8
12	1	0.1793	1

Table 6.3: A predicted race. Placement is the actual placement of the horse. Pred. Probability is the probability to win predicted by Model 3 and Ranked odds is how the odds ranked the horses.

From the ranks it can be seen that the result of the model is quite similar to the results of the odds. By comparing table 6.3 with table 6.2 it is easy to see that model 2 performs slightly better than model 3.

6.3 Betting

Results of betting on V75, with the condition that the cost must not exceed 500 SEK, according to the win often betting strategy using model two is shown in table 6.4. Usually money is won if at least five of the winners are included in the system. However if there are plenty of systems with five or six of the winners no one will get any money for these, instead they are saved in a jackpot until the next V75 day.

V75	Num. winners	Cost	Won	Sum
1	2	480	0	−480
2	6	480	*Jackpot*	−960
3	3	500	0	−1460
4	6	480	48	−1892
5	7	486	1966	−412
6	5	480	*Jackpot*	−912
7	5	480	*Jackpot*	−1372

Table 6.4: V75 with model 2. The Num. winners column display how many of the actual winners that were included in the system. The cost column display the cost of the system while the Won column displays money won on the system. The Sum column display the account balance.

The same V75 days are predicted but this time with model three, the result is shown in table 6.5.

V75	Num. winners	Cost	Won	Sum
1	4	486	0	−486
2	6	486	*Jackpot*	−972
3	3	500	0	−1472
4	5	500	*Jackpot*	−1972
5	5	486	*Jackpot*	−2458
6	6	486	21	−2923
7	5	486	*Jackpot*	−3409

Table 6.5: V75 with model 3.The Num. winners column display how many of the actual winners that were included in the system. The cost column display the cost of the system while the Won column displays money won on the system. The Sum column display the account balance.

What can be said from these two tables is that the second model appears to be better. However this strategy does not seem to be profitable. When the number of winners are close to seven the return is usually not big, i.e. a lot of other players have included the same or more winners in their systems. The reason for this is that the prediction from the models does not usually differ a lot from the prediction of the odds.

To investigate whether it would be profitable to bet according to the Betting in a Single Race strategy in section 5.2 with our selected model, a bet of 10 SEK is placed in each race if any horse j in the race satisfies the requirement

$$p_j > p_{odds_j}, \tag{6.2}$$

i.e. the predicted probability that a horse j should win the race has to be greater than the probability from the odds otherwise no bet will be placed on it. This is combined with the criteria in equation 5.10 that $p_j \geq \mu_1 = 0.1$, i.e. the probability also has to be bigger than 0.1. Though if several horses in the same race meets this condition, the bet is placed on the horse with the greatest predicted probability of winning. Additional if a horse j whom meets these conditions also has a large predicted probability, let's say $p_j \geq \mu_2 = 0.2$, it should have a good chance of winning and a bet of 20 SEK will be placed instead. If no horse meets these conditions no bet will be placed in that race.

In figure 6.3.1 the result is shown. Starting with a bankroll of 100 SEK, after 68 races the bankroll has grown to 285 SEK, or in percentage, with 185% using model two and with model three the bankroll has grown to 163 SEK i.e. with 63%.

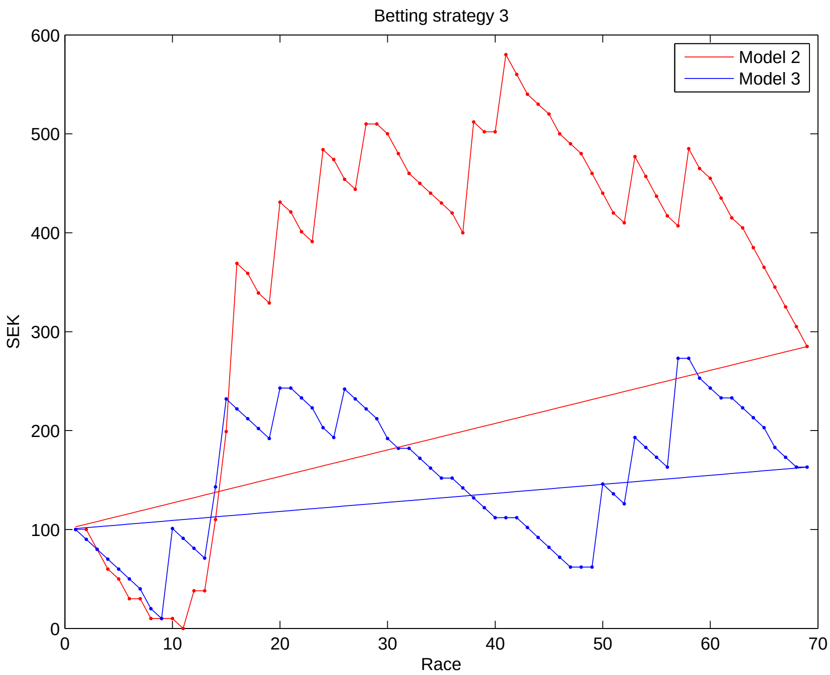

Figure 6.3.1: Result with betting strategy 3. Both models will result in a increase of the bankroll. Also notice that the large increase of the bankroll at the left of the plot is not just one but several wins.

This indicates that betting according to this strategy using any of these models will increase your bankroll. However betting based on model three seems more stable but betting based on model two more profitable. As the number of predicted races is not as large as what would have been preferred this result is still a bit unsure. If the number of predicted races would have been increased by two or three times perhaps a different pattern could have been seen.

Chapter 7

Discussion

7.1 Thoughts

The models and the first betting strategy is constructed to win as often as possible. This means that probably all of the favorites, based on the odds, will be included in the optimized system. Therefore, using one of the models with this betting strategy in a money earning purpose might not be successful. The downside is the return. Though betting on V75 using the strategy based on reduced systems, see Appendix A, might be more successful. The strategy where a bet is places on a single horse seem to be a better option if making money is the only goal. Before using any of the models for betting according to this strategy some further testing is necessary.

If data would have been easier to access the best way to create a model would probably be to only include V75 races. This would result in a data set where no, or at least very few, horses with a small amount of data would be present. This would most likely result in a more suitable model for predicting V75 races. Unfortunately, this was not tested due to the block making it impossible to get all data on time.

7.2 For the Future

Since the model is performing quite well already and there are some improvements which could make it even better, see section 7.1, the next step in developing the model is to collect this data and run a new regression. Hopefully this will result in an even better model. Also, due to the fact that `travsport.se` removes foreign registered horses a few days after a race, future races should be saved immediately resulting in a bigger and better data set which can be used either for running regressions or testing the predictive ability of the model.

Bibliography

[1] Svensk Travsport. *Tävlingsreglementet*. 2013.

[2] Madsen H. *Time Series Analysis*. Chapman & Hall/CRC. 2008.

[3] Lang H. *Topics on Applied Mathematical Statistics.* KTH. 2012.

[6] AB Trav och Galopp. www.atg.se. 2013.

[7] Hallgren J. *A Non-Parametric Approach to Finding the Probability Distribution Induced by a Set of Expected Positions.* Work in progress. 2013.

[8] Shuhua H. *Akaike Information Criterion.* North Carolina State University. 2007.

[9] Fawcett T. *An Introduction to ROC Analysis.* Elsevier. 2005

Appendix A

Betting Strategies

A.1 The Win Big Betting Strategy

A.1.1 Reduced Systems

If the aim is to win big, i.e. to maximise the chance of winning not so often but big when winning it might be suitable to bet according to a well-known betting strategy called "reduced systems". The main purpose with betting like this is to reduce the cost by not including combinations that won't give back a big return. Therefore several different betting systems are made where each system only includes, for example, two or three favorites. These kind of criteria can be chosen in different ways. One way is to rank the horses as A-,B- and C-horses depending on how good the horses are considered. A-horses being the best ones and C-horses being the less good, i.e. those who can make the return really big. As mentioned, this is a well known betting strategy and several programs to rank the horses and then create a reduced system already exist. So to reduce the risk of betting like others a new and more unique strategy based on reduced systems was developed. Let's call it Reduced Systems 2.0

A.1.2 Reduced Systems 2.0

To distinguish this strategy from the other, more well-known, strategies a new way of creating the reduced system has to be created. The idea is to put a constraint on the lowest accepted odds. This means that the system will not include the rows for which the odds is considered too low and therefore make sure that the return will be quite big if one of the rows is correct. When all of the rows which satisfy the constraint are known it is possible to find the ones for which the probability to be correct is highest. This is easily done in MATLAB by simply calculating the probability of a row being correct and then repeating this for all of the rows and then pick out the ones with highest probabilities to be correct. How many rows that will be used depends on how much money the player wants to bet. For example a bet of 500 SEK is equivalent to 1000 rows.

Reduced Systems 2.0 with more constraints

Depending on how difficult to predict a certain game day is the amount of rows which satisfy the constraint on the odds can vary a lot. There is a very simple way to reduce this amount a lot, namely to add more constraints. Though this time the constraint are, for example, that a certain horse must win its race. This reduces the amount of possible rows only to be the rows where this horse wins. Since the horses are ranked by the model this makes the new constraints more like demands on the predictive ability of model. The disadvantage of putting new constraints like the one mentioned is of course that if it the constraint is not satisfied the whole reduced system fails. So before adding new constraints it is necessary to analyse the races thoroughly.

Appendix B

MATLAB Scripts - Data

```
%% Script to find the id of horses a V75 race day

clc, close all, clear all
tic

URLvec = []; antalhorses = []; horsevec = [];

tevdagID='529896'; % A given race day
url= ['https://www.travsport.se/sresultat?kommando=
  tevlingsdagVisa&tevdagId=' tevdagID];

urlwrite(url, 'test.txt');

filnamn = ['test.txt'];
fid = fopen(filnamn,'rt');
C=textread(filnamn, '%s', 'delimiter', '\n');

j = 1; n = 1;
hitta = 'V75-1'; % Find every V75 race

while j < length(C)
    rad = C{j};

    if strfind(rad, hitta)
        j = j + 1;
        rad = C{j};
        start = length('<a href="') + 1;
        stop = strfind(rad, '" class="large" >') - 1;
        n = n + 1;
        hitta = ['V75-' num2str(n)];
        URLvec = [URLvec; {num2str(rad(start:stop))}];
    elseif strfind(rad, 'Strukna')
        j=length(C);
```

```
    end
    j = j+1;
end
fclose(fid);

% Find the horses
hitta = '<a href="/hast/visa/';

for i = 1:length(URLvec)
    url = ['https://www.travsport.se' URLvec{i}];
    pause(1)
    urlwrite(url, 'test.txt');
    fid = fopen(filnamn,'rt');
    C=textread(filnamn, '%s', 'delimiter', '\n');
    n = 0;
    j = 1;
    while j < length(C)
     rad = C{j};
        if strfind(rad, hitta)
            start = length(hitta) + 1;
            stop = strfind(rad, '/resultat"') - 1;
            horsevec = [horsevec; {(rad(start:stop))}];
            n = n + 1;
        elseif strfind(rad, 'Strukna')
        j=length(C);
        end
        j = j + 1;
    end
    fclose(fid);

    antalhorses = [antalhorses n];
end
save('antalhorses','antalhorses');
save('horsevec','horsevec'); %Saves a vector of horses
  participating that day

delete('C:\Users\mhella\Documents\MATLAB\KEX\NY\test.txt');
InmatningTillModeleringsprogrammet
loppinfo2_martinTillmodelleringINKLutlandskalopp
toc
```

```
%% Script to download HMTL from travsport.se

load('horsevec') %A given vector of horses

for n = 1:length(horsevec)
    filnamn = [num2str(horsevec{n}) '.txt'];
    url= ['https://www.travsport.se/hast/visa/' num2str(
      horsevec{n}) '/resultat'];
```

```
    try
        bla = urlwrite(url, filnamn);
    catch err
        disp(num2str(n)) %Catches error if theres no horse
          with that id
    end
end
```

```
%% Textfile to struct

clc, close all, clear all
tic
hej = 0;

% STRUCT
horse.namn = [];horse.id = [];horse.color = [];horse.gender
  = [];
horse.birthdate = [];horse.ras = [];horse.loppid = [];horse
  .datum = [];
horse.startnummer = [];horse.loppnummer = [];horse.distans
  =[];
horse.tid = [];horse.resultat=[];horse.ntungbana = [];horse
  .odds = [];
horse.kusk = [];horse.trainer = [];horse.vinterbana = [];
horse.tungbana = [];horse.autostart = [];horse.galopp = [];
horse.skorfram = [];horse.skorbak = [];horse.vinst = [];
  horse.bana=[];
horse_id = [];

load('horsevec'); %Loading vector of downloaded horses

ok = 1;
i = 1;

for h = 1:length(horsevec)
    n = horsevec{h};
    check = 1;
    try
        filnamn = [num2str(n) '.txt'];
        fid = fopen(filnamn,'rt');
        C=textread(filnamn, '%s', 'delimiter', '\n');
    catch err
        check = 0;
    end
    if check == 1
        for j = 1:length(C)
            rad = C{j};
            okej4=1;

```

```
% Name
if j == 460
    start = length('<span class="notranslate">')+1;
    stop = strfind(rad, '</span> <br/>') - 1;
horse(i).namn = rad(start:stop);

    % Color
elseif j == 465
    start = 5;
    stop = strfind(rad, '</td>') - 1;
    horse(i).color = rad(start:stop);

    % Gender
elseif j == 466
    start = 14;
    stop = strfind(rad, '</td>') - 1;
    horse(i).gender = rad(start:stop);

    % Birthdate
elseif j == 467
    horse(i).birthdate = str2double([rad(14:17) rad(19:20) rad(22:23)]);

    % Race
elseif j == 468
    start = 5;
    stop = strfind(rad, '</td>') - 1;
    horse(i).ras = rad(start:stop);
end

if isempty(strfind(rad, '130504')) == 0
    ok = 1;

    % Racetrack
    radbana=C{j-1};
    start = 5;
    stop = strfind(radbana, '</td>') - 1;
    horse(i).bana = [horse(i).bana; {radbana(start:stop)}];
    radstart=C{j+2};

    % Starting number
    start = length('<td colspan="2" class="right_align"><span>') + 1;
    slash = strfind(radstart(start:end), '/');
    stop = start - 1 + slash(1) - 1;
    horse(i).startnummer = [horse(i).startnummer; str2double(radstart(start:
```

```
                stop))];

            % Date
            horse(i).datum= [horse(i).datum; 130504];

            % Distance
            start = stop + 2;
            stop = start - 1 + strfind(radstart(start:
                end), '</span></td>') - 1;
            horse(i).distans= [horse(i).distans;
                str2double(radstart(start:stop))];

            % Driver
            radkusk = C{j+14};
            kuskstart = strfind(radkusk,'>') + 1;
            kuskstop = strfind(radkusk,'</') - 1;
            horse(i).kusk = [horse(i).kusk; {radkusk(
                kuskstart:kuskstop)}];

            horse(i).loppid = [horse(i).loppid; 0];
            horse(i).resultat = [horse(i).resultat; 0];
            horse(i).loppnummer = [horse(i).loppnummer;
                 0];
            horse(i).tid = [horse(i).tid; 0];
            horse(i).ntungbana = [horse(i).ntungbana;
                0];
            horse(i).trainer = [horse(i).trainer; {0}];
            horse(i).skorbak = [horse(i).skorbak; 0];
            horse(i).skorfram = [horse(i).skorfram; 0];
            horse(i).vinst = [horse(i).vinst; 0];
            horse(i).galopp = [horse(i).galopp; 0];
            horse(i).tungbana = [horse(i).tungbana; 0];
            horse(i).vinterbana = [horse(i).vinterbana;
                 0];
            horse(i).autostart = [horse(i).autostart;
                0];
        end

        if isempty(strfind(rad, 'loppId')) == 0
            ok = 1;

            % Race id
            start = strfind(rad, 'loppId') + 1 + length
                ('loppId');
            stop = start - 1 + strfind(rad(start:end),
                '"><span>') - 1;
            horse(i).loppid = [horse(i).loppid;
                str2double(rad(start:stop))];

```

```
            % Racetrack
            radbana=C{j-1};
            start = 5;
            stop = strfind(radbana, '</td>') - 1;
            horse(i).bana = [horse(i).bana; {radbana(start:stop)}];

            % Date
            start = stop + length('"><span>') + 1;
            stop = start - 1 + strfind(rad(start:end), '-') - 1;
            horse(i).datum = [horse(i).datum; str2double(rad(start:stop))];

            % LOPPNUMMER
            start = stop + 2;
            stop = start - 1 + strfind(rad(start:end), '</span') - 1;
            horse(i).loppnummer = [horse(i).loppnummer; str2double(rad(start:stop))];

        elseif isempty(strfind(rad, '<td class="nowrap"><span><em><span>')) == 0
            ok = 1;

            % Race id
            horse(i).loppid = [horse(i).loppid; 0];

            % Date
            start = strfind(rad, '<td class="nowrap"><span><em><span>') + length('<td class="nowrap"><span><em><span>');
            stop = strfind(rad, '-') - 1;
            horse(i).datum = [horse(i).datum; str2double(rad(start:stop))];

            % Race number
            start = stop + 2;
            stop = start - 1 + strfind(rad(start:end), '</span></em></span></td>') - 1;
            horse(i).loppnummer = [horse(i).loppnummer; str2double(rad(start:stop))];
        end

        if isempty(strfind(rad, '<td colspan="2" class="right_align"><span>')) == 0 && length(horse(i).datum) >              length(horse(i).startnummer)

```

```
% Result
radresultat = C{j+1};
resultatindex = strfind(radresultat, '>') + 1;
resultat = radresultat(resultatindex(1));

if isnan(str2double(resultat)) == 1 && strcmp(resultat, 'd')==0
    ok = 0;
    horse(i).loppid(end) = [];
    horse(i).datum(end) = [];
    horse(i).loppnummer(end) = [];
elseif isnan(str2double(resultat)) == 1 && strcmp(resultat, 'd') == 1
    hej = hej+1;
    horse(i).resultat = [horse(i).resultat; 0];
else
    horse(i).resultat = [horse(i).resultat; str2double(resultat)];
end

if ok == 1
    %horse(i).resultat = [horse(i).resultat; resultat];

    % Starting number
    start = length('<td colspan="2" class="right_align"><span>') + 1;
    slash = strfind(rad(start:end), '/');
    stop = start - 1 + slash(1) - 1;
    horse(i).startnummer = [horse(i).startnummer; str2double(rad(start:stop))];

    % Distance
    start = stop + 2;
    stop = start - 1 + strfind(rad(start:end), '</span></td>') - 1;

    % Track status
    if strcmp(rad(stop), 'n')
        stop = stop - 1;
        horse(i).ntungbana = [horse(i).ntungbana; 1];
        horse(i).vinterbana = [horse(i).vinterbana; 0];
        horse(i).tungbana = [horse(i).tungbana; 0];
```

```
            elseif strcmp(rad(stop), 'v')
                stop = stop - 1;
                horse(i).ntungbana = [horse(i).ntungbana; 0];
                horse(i).vinterbana = [horse(i).vinterbana; 1];
                horse(i).tungbana = [horse(i).tungbana; 0];
            elseif strcmp(rad(stop), 't')
                stop = stop - 1;
                horse(i).ntungbana = [horse(i).ntungbana; 0];
                horse(i).vinterbana = [horse(i).vinterbana; 0];
                horse(i).tungbana = [horse(i).tungbana; 1];
            else
                horse(i).ntungbana = [horse(i).ntungbana; 0];
                horse(i).vinterbana = [horse(i).vinterbana; 0];
                horse(i).tungbana = [horse(i).tungbana; 0];
            end
            horse(i).distans = [horse(i).distans; str2double(rad(start:stop))];

            % Time
            radtid = C{j+3};
            tidstop = strfind(radtid,'/');

            % Galop and car start
            if strcmp(radtid(tidstop-2), 'g') == 1 && strcmp(radtid(tidstop-3), 'a') == 1
                horse(i).galopp = [horse(i).galopp; 1];
                horse(i).autostart = [horse(i).autostart; 1];
                tidstop=tidstop-4;
            elseif strcmp(radtid(tidstop-2), 'g') == 1
                horse(i).galopp = [horse(i).galopp; 1];
                horse(i).autostart = [horse(i).autostart; 0];
                tidstop=tidstop-3;
            elseif strcmp(radtid(tidstop-2), 'a') == 1
```

```
    horse(i).autostart = [horse(i).autostart; 1];
    horse(i).galopp = [horse(i).galopp; 0];
    tidstop=tidstop-3;
else
    horse(i).autostart = [horse(i).autostart; 0];
    horse(i).galopp = [horse(i).galopp; 0];
    tidstop=tidstop-2;
end
tid=[radtid(5:tidstop-2) '.' radtid(tidstop-1:tidstop)];
horse(i).tid = [horse(i).tid; str2double(tid)];

% Odds
radodds = C{j+4};
oddsstop = strfind(radodds, '</span></td>');
odds = radodds(31:oddsstop-1);
horse(i).odds = [horse(i).odds; str2double(odds)];

% Shoes
radsko = C{j+8};
skostop = strfind(radsko,'gif"');
if isempty(skostop) == 1
    horse(i).skorfram = [horse(i).skorfram; 0];
    horse(i).skorbak = [horse(i).skorbak; 0];
else
    if strcmp(radsko(skostop(1)-5), '/') == 1
        horse(i).skorfram = [horse(i).skorfram; 1];
    else
        horse(i).skorfram = [horse(i).skorfram; 0];
    end
    if strcmp(radsko(skostop(2)-5), '/') == 1
        horse(i).skorbak = [horse(i).skorbak; 1];
    else
        horse(i).skorbak = [horse(i).skorbak; 0];
```

```
        end
    end

    % Driver
    radkusk = C{j+12};
    kuskstart = strfind(radkusk,'>') + 1;
    kuskstop = strfind(radkusk,'</') - 1;
    horse(i).kusk = [horse(i).kusk; {radkusk(kuskstart:kuskstop)}];

    % Trainer
    radtrainer = C{j+16};
    trainerstart = strfind(radtrainer,'>') + 1;
    trainerstop = strfind(radtrainer,'</') - 1;
    horse(i).trainer = [horse(i).trainer; {radtrainer(trainerstart:trainerstop)}];

    % Money
    radvinst = C{j+20};
    start = length('<td class="right_align nowrap"><span>') + 1;
    stop = strfind(radvinst, '</span></td>') - 1;
    vinst = radvinst(start:stop);
    if isempty(strfind(vinst, ' ')) == 0
        blank = strfind(vinst, ' ');
        vinst(blank) = [];
    end
    horse(i).vinst = [horse(i).vinst; str2double(vinst)];
end

if isnan(horse(i).odds) == 1
    horse(i).loppid(end) = [];
    horse(i).datum(end) = [];
    horse(i).startnummer(end) = [];
    horse(i).loppnummer(end) = [];
    horse(i).distans(end) = [];
    horse(i).tid(end) = [];
    horse(i).resultat(end) = [];
    horse(i).ntungbana(end) = [];
    horse(i).odds(end) = [];
    horse(i).kusk(end) = [];
    horse(i).trainer(end) = [];
    horse(i).vinterbana(end) = [];
    horse(i).tungbana(end) = [];
```

```
                    horse(i).autostart(end) = [];
                    horse(i).galopp(end) = [];
                    horse(i).skorfram(end) = [];
                    horse(i).skorbak(end) = [];
                    horse(i).vinst(end) = [];
                    horse(i).bana(end) = [];
                end
            end
        end

        if isempty(horse(i).loppid) == 0
            horse_id = [horse_id; n];
            horse(i).id = n;
            i = i + 1;
        else
        horse(end)=[];
        end
        fclose(fid);
    end
end

save('horse','horse');
toc
```

```
%% Creating a single struct with new variables from several
   structs

loppdatum=121012; %No races after this date will be
  included

load('longhorse') %Load file

namn = []; id =[]; alder = [];segerform = [];Dtemp = [];
banatrivselindex = 1;distanstrivselindex = 1;
  autotrivselindex = 1;
sommartrivselindex = 1;

redhorse = horse2redhorse(horse,loppdatum); %Function

for j =1:length(redhorse)
    id = [id; redhorse(j).id];
end

antal = length(id);

birthdate = zeros(antal,1);ras = zeros(antal,1);loppid =
  zeros(antal,1);datum = zeros(antal,1);
startnummer = zeros(antal,1);loppnummer = zeros(antal,1);
  distans=zeros(antal,1);
```

```
tid = zeros(antal,1);resultat=zeros(antal,1);ntungbana = zeros(antal,1);odds = zeros(antal,1);
vinterbana = zeros(antal,1);
tungbana = zeros(antal,1); autostart = zeros(antal,1); galopp = zeros(antal,1);
skorfram = zeros(antal,1);skorbak = zeros(antal,1);vinst = zeros(antal,1);
kusk = {zeros(antal,1)}; trainer = {zeros(antal,1)}; bana = {zeros(antal,1)};
gender = {zeros(antal,1)};

start = 1;startpindex = 1;vinstformindex = 1;segerformindex =1;

vinstform = zeros(antal,1);
startp = zeros(antal,1);
trivselfaktor1 = zeros(antal,1);
banavinsttrivsel = zeros(antal,1);
banatopp3trivsel = zeros(antal,1);
distansvinsttrivsel = zeros(antal,1);
distanstopp3trivsel = zeros(antal,1);
autovinsttrivsel = zeros(antal,1);
autotopp3trivsel = zeros(antal,1);
sommarvinsttrivsel = zeros(antal,1);
sommartopp3trivsel = zeros(antal,1);

for j=1:length(redhorse)
    stop = start + length(redhorse(j).id) - 1;
    namn = [namn; {redhorse(j).namn}];
    resultat(start:stop) = redhorse(j).resultat;
    startnummer(start:stop) = redhorse(j).startnummer;
    loppnummer(start:stop) = redhorse(j).loppnummer;
    distans(start:stop) = redhorse(j).distans;
    tid(start:stop) = redhorse(j).tid;
    ntungbana(start:stop) = redhorse(j).ntungbana;
    vinterbana(start:stop) = redhorse(j).vinterbana;
    tungbana(start:stop) = redhorse(j).tungbana;
    odds(start:stop) = redhorse(j).odds;
    kusk(start:stop) = redhorse(j).kusk;
    trainer(start:stop) = redhorse(j).trainer;
    autostart(start:stop) = redhorse(j).autostart;
    galopp(start:stop) = redhorse(j).galopp;
    skorfram(start:stop) = redhorse(j).skorfram;
    skorbak(start:stop) = redhorse(j).skorbak;
    vinst(start:stop) = redhorse(j).vinst;
    bana(start:stop) = redhorse(j).bana;
    datum(start:stop) = redhorse(j).datum;
    loppid(start:stop) = redhorse(j).loppid;
        for r = start:stop
```

```
            gender(r) = {redhorse(j).gender};
        end

    start = stop + 1;

    % Age
    fodd = num2str(horse(j).birthdate);
    alder = [alder; round(redhorse(j).datum/10^4)-round(str2double(fodd(3:8))/10^4)];

    % Money shape
    [vinstform, vinstformindex] = vinstformfunc(redhorse(j),vinstform,vinstformindex);

    % Win shape
    [segerform, segerformindex] = segerformfunc(redhorse(j),segerform,segerformindex);

    % Starting point
    [startp, startpindex] = startpfunc(redhorse(j),startp,startpindex);

    % Distance/track/starting/season fit
    won = find(redhorse(j).resultat == 1);
    second = find(redhorse(j).resultat == 2);
    third = find(redhorse(j).resultat == 3);
    topp3 = [won;second;third];
    if isempty(won) == 0
        vinstprocent = length(won)/length(redhorse(j).resultat);
    else
        vinstprocent = 0;
    end
    if isempty(topp3) == 0
        topp3procent = length(topp3)/length(redhorse(j).resultat);
    else
        topp3procent = 0;
    end

    if isempty(redhorse(j).resultat) == 0

        % Track fit
        [banavinsttrivsel, banatopp3trivsel, banatrivselindex] = trivselbana(banavinsttrivsel, banatopp3trivsel, banatrivselindex,...
        redhorse(j), topp3, won, topp3procent, vinstprocent);

```

```
        % Distance fit
        [distansvinsttrivsel, distanstopp3trivsel, distanstrivselindex] =       trivseldistans(distansvinsttrivsel,distanstopp3trivsel,distanstrivselindex,...
        redhorse(j), topp3, won, topp3procent, vinstprocent);

        % Starting fit
        [autovinsttrivsel, autotopp3trivsel, autotrivselindex] =        trivselauto(autovinsttrivsel,autotopp3trivsel,autotrivselindex,...
        redhorse(j), topp3, won, topp3procent, vinstprocent);

        % Season fit
        [sommarvinsttrivsel, sommartopp3trivsel, sommartrivselindex] =      trivselsommar(sommarvinsttrivsel,   sommartopp3trivsel,sommartrivselindex,...
        redhorse(j), topp3, won, topp3procent, vinstprocent);
    end


end

% Driver rank
[kuskrank, brakusk, ganskabrakusk, mycketbrakusk] = kuskrankfunc(redhorse, resultat);

% Gender dummy
[valack, hingst, sto] = genderfunc(gender);

% Horse rank
[vinstprocent, platsprocent, mktbrahorse, brahorse, ganskabrahorse] =...
horserankfunc(redhorse,resultat);

% Money and time
[pengar, mintid] = pengar_mintid_func(redhorse,resultat);

% Dummy matrix
if j == 1
    dummie = sparse([zeros(length(redhorse(j).resultat), length(redhorse)-1)]);
else
```

```
    dummie = sparse([zeros(length(redhorse(j).resultat),j
      -2)...
    ones(length(redhorse(j).resultat),1) zeros(length(
      redhorse(j).resultat),...
    length(redhorse)-j)]);
end
Dtemp = [Dtemp; dummie];
etta = find(Dtemp == 1);
D = zeros(size(Dtemp));
D(etta) = 1;

superhorse.id = id;
superhorse.resultat = resultat;
superhorse.startnummer = startnummer;
superhorse.loppnummer = loppnummer;
superhorse.distans = distans;
superhorse.tid = tid;
superhorse.ntungbana = ntungbana;
superhorse.vinterbana = vinterbana;
superhorse.tungbana = tungbana;
superhorse.odds = odds;
superhorse.kusk = kusk;
superhorse.trainer = trainer;
superhorse.autostart = autostart;
superhorse.galopp = galopp;
superhorse.skorfram = skorfram;
superhorse.skorbak = skorbak;
superhorse.vinst = vinst;
superhorse.alder = alder;
superhorse.vinstform = vinstform;
superhorse.startp = startp;
superhorse.datum = datum;
superhorse.namn = namn;
superhorse.trivselfaktor = trivselfaktor1;
superhorse.brakusk = brakusk;
superhorse.ganskabrakusk = ganskabrakusk;
superhorse.kuskrank = kuskrank;
superhorse.banavinsttrivsel = banavinsttrivsel;
superhorse.banatopp3trivsel = banatopp3trivsel;
superhorse.distansvinsttrivsel = distansvinsttrivsel;
superhorse.distanstopp3trivsel = distanstopp3trivsel;
superhorse.autovinsttrivsel = autovinsttrivsel;
superhorse.autotopp3trivsel = autotopp3trivsel;
superhorse.sommarvinsttrivsel = sommarvinsttrivsel;
superhorse.sommartopp3trivsel = sommartopp3trivsel;
superhorse.valack = valack;
superhorse.hingst = hingst;
superhorse.sto = sto;
superhorse.mycketbrakusk = mycketbrakusk;
```

```
superhorse.mktbrahorse = mktbrahorse;
superhorse.brahorse = brahorse;
superhorse.ganskabrahorse = ganskabrahorse;
superhorse.vinstprocent = vinstprocent;
superhorse.pengar = pengar;
superhorse.loppid = loppid;
superhorse.platsprocent = platsprocent;
superhorse.segerform = segerform;
superhorse.mintid=mintid;

& Starting number rank, car start
RankA=tiedrank(-[0.1823 0.1910 0.2024 0.2261 0.2386 0.1877
  0.1583 0.1312 0.1288...
0.1369 0.1542 0.1172 0.1538 0.1176 0.1400]);

% Starting number rank, volt start
RankV=tiedrank(-[0.2289 0.2036 0.1963 0.1528 0.1653 0.2151
  0.1991 0.1510 0.1472...
0.1362 0.1426 0.1214]);

StartnummerRank=[];
for n=1:length(superhorse.startnummer)
    if isnan(superhorse.startnummer(n))==1
        superhorse.startnummer(n)=randi([1, 12]);
    end
    if superhorse.autostart(n) == 1;
        StartnummerRank=[StartnummerRank; RankA(superhorse.
          startnummer(n))];
    else
        StartnummerRank=[StartnummerRank; RankV(superhorse.
          startnummer(n))];
    end
end

n=1;
% Placement
omvresultat = zeros(length(superhorse.resultat),1);
for n=1:length(superhorse.resultat)
    if superhorse.resultat(n)==0
        omvresultat(n)=10;
    else
        omvresultat(n)=superhorse.resultat(n);
    end
end

superhorse.StartnummerRank = StartnummerRank;
superhorse.omvresultat = omvresultat;
```

```
function [valack, hingst, sto] = genderfunc(gender)
```

```

valack = zeros(length(gender), 1);
hingst = zeros(length(gender), 1);
sto = zeros(length(gender), 1);

index = 1;
for j = 1:length(gender)
    if strcmp(gender(j), 'valack') == 1
        valack(index) = 1;
    elseif strcmp(gender(j), 'hingst') == 1
        hingst(index) = 1;
    elseif strcmp(gender(j), 'sto') == 1
        sto(index) = 1;
    end
    index = index+1;
end
```

```
function redhorse = horse2redhorse(horse,loppdatum)
redhorse = struct;
redhorse.namn = [];
redhorse.id = [];
redhorse.color = [];
redhorse.gender = [];
redhorse.birthdate = [];redhorse.ras = [];redhorse.loppid = [];redhorse.datum = [];
redhorse.startnummer = [];redhorse.loppnummer = [];redhorse.distans=[];
redhorse.tid = [];redhorse.resultat=[];redhorse.ntungbana = [];redhorse.odds = [];
redhorse.kusk = [];redhorse.trainer = [];redhorse.vinterbana = [];
redhorse.tungbana = [];redhorse.autostart = [];redhorse.galopp = [];
redhorse.skorfram = [];redhorse.skorbak = [];redhorse.vinst = []; redhorse.bana = [];

s=0;
for j=1:length(horse)
    redhorse(j).namn = horse(j).namn;
    k = 1;
    while k <= length(horse(j).datum)
        if horse(j).datum(k) < loppdatum
            for i =k:length(horse(j).datum)
                redhorse(j).id = [redhorse(j).id; horse(j).id];
            end

        redhorse(j).resultat = [redhorse(j).resultat; horse(j).resultat(k:end)];
```

```
        redhorse(j).startnummer = [redhorse(j).startnummer; horse(j).startnummer(k:end)];
        redhorse(j).loppnummer = [redhorse(j).loppnummer; horse(j).loppnummer(k:end)];
        redhorse(j).distans = [redhorse(j).distans; horse(j).distans(k:end)];
        redhorse(j).tid = [redhorse(j).tid; horse(j).tid(k:end)];
        redhorse(j).ntungbana = [redhorse(j).ntungbana; horse(j).ntungbana(k:end)];
        redhorse(j).vinterbana = [redhorse(j).vinterbana; horse(j).vinterbana(k:end)];
        redhorse(j).tungbana = [redhorse(j).tungbana; horse(j).tungbana(k:end)];
        redhorse(j).odds = [redhorse(j).odds; horse(j).odds(k:end)];
        redhorse(j).kusk = [redhorse(j).kusk; horse(j).kusk(k:end)];
        redhorse(j).trainer = [redhorse(j).trainer; horse(j).trainer(k:end)];
        redhorse(j).autostart = [redhorse(j).autostart; horse(j).autostart(k:end)];
        redhorse(j).galopp = [redhorse(j).galopp; horse(j).galopp(k:end)];
        redhorse(j).skorfram = [redhorse(j).skorfram; horse(j).skorfram(k:end)];
        redhorse(j).skorbak = [redhorse(j).skorbak; horse(j).skorbak(k:end)];
        redhorse(j).vinst = [redhorse(j).vinst; horse(j).vinst(k:end)];
        redhorse(j).datum = [redhorse(j).datum; horse(j).datum(k:end)];
        redhorse(j).bana = [redhorse(j).bana; horse(j).bana(k:end)];
        redhorse(j).gender = horse(j).gender;
        redhorse(j).loppid = [redhorse(j).loppid; horse(j).loppid(k:end)];

        nanvektor = find(isnan(redhorse(j).odds));
        if isempty(nanvektor) == 0
            nanvektor = fliplr(nanvektor');
                for m = 1:length(nanvektor')
                    s=s+1;
                    n = nanvektor(m);

                    redhorse(j).id(end) = [];
                    redhorse(j).resultat(n) = [];
                    redhorse(j).startnummer(n) = [];
                    redhorse(j).loppnummer(n) = [];
```

```
                    redhorse(j).distans(n) = [];
                    redhorse(j).tid(n) = [];
                    redhorse(j).ntungbana(n) = [];
                    redhorse(j).vinterbana(n) = [];
                    redhorse(j).tungbana(n) = [];
                    redhorse(j).odds(n) = [];
                    redhorse(j).kusk(n) = [];
                    redhorse(j).trainer(n) = [];
                    redhorse(j).autostart(n) = [];
                    redhorse(j).galopp(n) = [];
                    redhorse(j).skorfram(n) = [];
                    redhorse(j).skorbak(n) = [];
                    redhorse(j).vinst(n) = [];
                    redhorse(j).datum(n) = [];
                    redhorse(j).bana(n) = [];
                    redhorse(j).loppid(n) = [];
                end
            end
            k = length(horse(j).datum);
        end
        k = k+1;
    end
end
end
```

```
function [vinstprocent, platsprocent, mktbrahorse, brahorse, ganskabrahorse] =...
horserankfunc(redhorse,resultat)

vinstprocent = zeros(length(resultat),1);
platsprocent = zeros(length(resultat),1);
antalloppvec = zeros(length(resultat),1);
mktbrahorse = zeros(length(resultat),1);
brahorse = zeros(length(resultat),1);
ganskabrahorse = zeros(length(resultat),1);
pengar = zeros(length(resultat),1);
idhorse = zeros(length(resultat),1);
mintid = zeros(length(resultat),1);

n=1;
for j=1:length(redhorse)
    antallopptot = length(redhorse(j).resultat);
    for i=2:antallopptot
        distans1 = redhorse(j).distans(i-1);
        index = find(redhorse(j).distans(i:end)==distans1);
        if isempty(index) == 1
            mintid(n) = 0;
        else
            mintid(n) = min(redhorse(j).tid(index));
```

```
            if isnan(mintid(n))==1
                mintid(n) = 0;
            end
        end

        antallopp = length(redhorse(j).resultat(i:end));
        antalvinster = length(find(redhorse(j).resultat(i:
          end) == 1));
        antalplatser = antalvinster + length(find(redhorse(
          j).resultat(i:end) == 2)) +       length(find(
          redhorse(j).resultat(i:end) == 3));
        pengar(n) = sum(redhorse(j).vinst(i:end))/antallopp
          ;
        vinstprocent(n) = antalvinster/antallopp;
        platsprocent(n) = antalplatser/antallopp;
        antalloppvec(n) = antallopp;

        idhorse(n) = redhorse(j).id(i);
        if vinstprocent(n) >= 0.3 && antallopp > 6
            mktbrahorse(n)=1;
        elseif vinstprocent(n) >= 0.20
            brahorse(n)=1;
        elseif vinstprocent(n) >= 0.15
            ganskabrahorse(n)=1;
        end
        n = n+1;
    end

    if antallopptot > 0
    n = n+1;
end
end
```

```
function [kuskrank, brakusk, ganskabrakusk, mycketbrakusk]
  = kuskrankfunc(redhorse, resultat)

load('kuskar.mat');
load('kuskprocent');

kuskrank = zeros(length(resultat),1);
brakusk = zeros(length(resultat),1);
ganskabrakusk = zeros(length(resultat),1);
mycketbrakusk = zeros(length(resultat),1);

n=1;
for j=1:length(redhorse)
    for i=1:length(redhorse(j).resultat)
        plats=find(strcmp(kuskar,redhorse(j).kusk(i)) == 1)
          ;
```

```
        if isempty(plats)==1
            mycketbrakusk(n)=0;
            brakusk(n)=0;
            ganskabrakusk(n)=0;
            n=n+1;
        else
            kuskrank(n)=kuskprocent(plats);
            if kuskprocent(plats)>0.2 && kuskprocent(plats)
              <0.32
                mycketbrakusk(n)=1;
                brakusk(n)=0;
                ganskabrakusk(n)=0;
                n=n+1;
            elseif kuskprocent(plats)>0.16 && kuskprocent(
              plats)<=0.2
                mycketbrakusk(n)=0;
                brakusk(n)=1;
                ganskabrakusk(n)=0;
                n=n+1;
            elseif kuskprocent(plats)>0.13 && kuskprocent(
              plats)<=0.16
                mycketbrakusk(n)=0;
                brakusk(n)=0;
                ganskabrakusk(n)=1;
                n=n+1;
            else
                mycketbrakusk(n)=0;
                brakusk(n)=0;
                ganskabrakusk(n)=0;
                n=n+1;
            end
        end
    end
end
```

```
function [pengar, mintid] = pengar_mintid_func(redhorse,
  resultat)

pengar = zeros(length(resultat),1);
mintid = zeros(length(resultat),1);

n=1;
for j=1:length(redhorse)
    antallopptot = length(redhorse(j).resultat);
        for i=2:antallopptot
            if redhorse(j).tid(i)<=9
                redhorse(j).tid(i)=NaN;
            end
        end
```

```
    for i=2:antallopptot
        k=redhorse(j).autostart(i-1);
        antallopp = length(redhorse(j).resultat(i:end));
        distans1 = redhorse(j).distans(i-1);
        index = find(redhorse(j).distans(i:end)>=(distans1
          -60) & redhorse(j).autostart(i:end)==k &
          redhorse(j).distans(i:end)<=(distans1+60));
        index = index + i - 1;

        if isempty(index) == 1
            mintid(n) = 0;
        else
            mintid(n) = min(redhorse(j).tid(index));
            if isnan(mintid(n))==1
            mintid(n) = 0;
            end
        end

    pengar(n) = sum(redhorse(j).vinst(i:end))/antallopp;
    n = n+1;
end

if antallopptot > 0
    n = n+1;
end
end
```

```
function [startp, startpindex] = startpfunc(redhorse,startp
  , startpindex)

startstartp = 1;
stopstartp = length(redhorse.vinst);

if length(redhorse.vinst) > 5
    startstartp = 1;
    stopstartp = length(redhorse.vinst);
    for k = startstartp:stopstartp-5
        startp_vinst = 0;
        for n = k + 1:k + 5
            if redhorse.resultat(n) == 1
                startp_vinst = startp_vinst + 300;
            elseif redhorse.resultat(n) == 2
                startp_vinst = startp_vinst + 150;
            elseif redhorse.resultat(n) == 3
                startp_vinst = startp_vinst + 100;
            elseif redhorse.resultat(n) == 4
                startp_vinst = startp_vinst + 50;
            elseif redhorse.resultat(n) == 5
```

```
                startp_vinst = startp_vinst + 25;
            end
        end

        startp(startpindex) = sum(redhorse.vinst(k+1:k+5))/100 + startp_vinst;
        startpindex = startpindex + 1;
    end

    for m =2:5
        startp_vinst = 0;
        for n = k + m:k + 5
            if redhorse.resultat(n) == 1
                startp_vinst = startp_vinst + 300;
            elseif redhorse.resultat(n) == 2
                startp_vinst = startp_vinst + 150;
            elseif redhorse.resultat(n) == 3
                startp_vinst = startp_vinst + 100;
            elseif redhorse.resultat(n) == 4
                startp_vinst = startp_vinst + 50;
            elseif redhorse.resultat(n) == 5
                startp_vinst = startp_vinst + 25;
            end
        end

        startp(startpindex) = sum(redhorse.vinst(k+m:k+5))/100 + startp_vinst;
        startpindex = startpindex + 1;
    end

    startp(startpindex) = 0;
    startpindex = startpindex + 1;

elseif length(redhorse.vinst) == 5
    for m =1:4
        startp_vinst = 0;

        for n = startstartp + m:startstartp + 4
            if redhorse.resultat(n) == 1
                startp_vinst = startp_vinst + 300;
            elseif redhorse.resultat(n) == 2
                startp_vinst = startp_vinst + 150;
            elseif redhorse.resultat(n) == 3
                startp_vinst = startp_vinst + 100;
            elseif redhorse.resultat(n) == 4
                startp_vinst = startp_vinst + 50;
            elseif redhorse.resultat(n) == 5
                startp_vinst = startp_vinst + 25;
            end
```

```
        end

        startp(startpindex) = sum(redhorse.vinst(
          startstartp+m:startstartp+4))/100 + startp_vinst;
        startpindex = startpindex + 1;
    end

    startp(startpindex) = 0;
    startpindex = startpindex + 1;

elseif length(redhorse.vinst) == 4
    for m =1:3
        startp_vinst = 0;
        for n = startstartp + m:startstartp + 3
            if redhorse.resultat(n) == 1
                startp_vinst = startp_vinst + 300;
            elseif redhorse.resultat(n) == 2
                startp_vinst = startp_vinst + 150;
            elseif redhorse.resultat(n) == 3
                startp_vinst = startp_vinst + 100;
            elseif redhorse.resultat(n) == 4
                startp_vinst = startp_vinst + 50;
            elseif redhorse.resultat(n) == 5
                startp_vinst = startp_vinst + 25;
            end
        end

        startp(startpindex) = sum(redhorse.vinst(
          startstartp+m:startstartp+3))/100 +
          startp_vinst;
        startpindex = startpindex + 1;
    end

    startp(startpindex) = 0;
    startpindex = startpindex + 1;

elseif length(redhorse.vinst) == 3
    for m =1:2
        startp_vinst = 0;
        for n = startstartp + m:startstartp + 2
            if redhorse.resultat(n) == 1
                startp_vinst = startp_vinst + 300;
            elseif redhorse.resultat(n) == 2
                startp_vinst = startp_vinst + 150;
            elseif redhorse.resultat(n) == 3
                startp_vinst = startp_vinst + 100;
            elseif redhorse.resultat(n) == 4
                startp_vinst = startp_vinst + 50;
            elseif redhorse.resultat(n) == 5
```

```
                startp_vinst = startp_vinst + 25;
            end
        end

        startp(startpindex) = sum(redhorse.vinst(
          startstartp+m:startstartp+2))/100 + startp_vinst;
        startpindex = startpindex + 1;
    end
    startp(startpindex) = 0;
    startpindex = startpindex + 1;

elseif length(redhorse.vinst) == 2
    startp_vinst = 0;
    for n = startstartp + 1:startstartp + 1
        if redhorse.resultat(n) == 1
            startp_vinst = startp_vinst + 300;
        elseif redhorse.resultat(n) == 2
            startp_vinst = startp_vinst + 150;
        elseif redhorse.resultat(n) == 3
            startp_vinst = startp_vinst + 100;
        elseif redhorse.resultat(n) == 4
            startp_vinst = startp_vinst + 50;
        elseif redhorse.resultat(n) == 5
            startp_vinst = startp_vinst + 25;
        end
    end

    startp(startpindex) = sum(redhorse.vinst(startstartp+1:
      startstartp+1))/100 + startp_vinst;
    startpindex = startpindex + 1;
    startp(startpindex) = 0;
    startpindex = startpindex + 1;

elseif length(redhorse.vinst) == 1
    startp(startpindex) = 0;
    startpindex = startpindex + 1;
end
```

```
function [segerform, segerformindex] = segerformfunc(
  redhorse,segerform, segerformindex)
tempres=redhorse.resultat;

for n=1:length(redhorse.resultat)
    if tempres(n)==0
        tempres(n)=10;
    end
end

if length(redhorse.vinst) > 3
```

```
    for k = 1:length(redhorse.vinst)-3
        segerform(segerformindex) = sum(tempres(k+1:k+3))
          /3;
        segerformindex = segerformindex+1;
    end

    segerform(segerformindex) = sum(tempres(k+2:k+3))/2;
    segerform(segerformindex + 1) = sum(tempres(k+3:k+3));
    segerform(segerformindex + 2) = 5;
    segerformindex = segerformindex+3;

elseif length(redhorse.vinst) == 3
    segerform(segerformindex) = sum(tempres(2:3))/2;
    segerform(segerformindex + 1) = sum(tempres(3:3));
    segerform(segerformindex + 2) = 5;
    segerformindex = segerformindex+3;

elseif length(redhorse.vinst) == 2
    segerform(segerformindex) = sum(tempres(2:2));
    segerform(segerformindex + 1) = 5;
    segerformindex = segerformindex+2;

elseif length(redhorse.vinst) == 1
    segerform(segerformindex) = 5;
    segerformindex = segerformindex+1;

else
end
```

```
function [vinstform, vinstformindex] = vinstformfunc(
  redhorse,vinstform, vinstformindex)

if length(redhorse.vinst) > 3
    for k = 1:length(redhorse.vinst)-3
        vinstform(vinstformindex) = sum(redhorse.vinst(k+1:
          k+3))/1000;
        vinstformindex = vinstformindex+1;
    end

    vinstform(vinstformindex) = sum(redhorse.vinst(k+2:k+3)
      )/1000;
    vinstform(vinstformindex + 1) = sum(redhorse.vinst(k+3:
      k+3))/1000;
    vinstform(vinstformindex + 2) = 0;
    vinstformindex = vinstformindex+3;

elseif length(redhorse.vinst) == 3
    vinstform(vinstformindex) = sum(redhorse.vinst(2:3))
      /1000;
```

```
    vinstform(vinstformindex + 1) = sum(redhorse.vinst(3:3))/1000;
    vinstform(vinstformindex + 2) = 0;
    vinstformindex = vinstformindex+3;

elseif length(redhorse.vinst) == 2
    vinstform(vinstformindex) = sum(redhorse.vinst(2:2))/1000;
    vinstform(vinstformindex + 1) = 0;
    vinstformindex = vinstformindex+2;

elseif length(redhorse.vinst) == 1
    vinstform(vinstformindex) = 0;
    vinstformindex = vinstformindex+1;
else
end
```

```
function [gvek, hvek, autotrivselindex] = trivselauto(gvek, hvek, autotrivselindex, redhorse, topp3, won, topp3procent, vinstprocent)

if length(redhorse.resultat) > 5
    vinstvec = zeros(length(redhorse.resultat),1);
    topp3vec = zeros(length(redhorse.resultat),1);
    resultat = redhorse.resultat;

    vinstprocentvec = zeros(length(redhorse.resultat),1);
    topp3procentvec = zeros(length(redhorse.resultat),1);
    vinstprocentavec=zeros(length(redhorse.resultat),1);
    topp3procentavec=zeros(length(redhorse.resultat),1);
    vinstprocentvvec=zeros(length(redhorse.resultat),1);
    topp3procentvvec=zeros(length(redhorse.resultat),1);

    index = find(resultat==1);
    vinstvec(index)=1;
    index = find(resultat >0 & resultat <4);
    topp3vec(index)=1;
    avec=redhorse.autostart;
    vvec = ones(length(redhorse.resultat),1)-avec;

    vinstavec=vinstvec.*avec;
    vinstvvec=vinstvec.*vvec;
    topp3avec=topp3vec.*avec;
    topp3vvec=topp3vec.*vvec;

    for k = 1:length(redhorse.resultat)
        vinstprocentvec(k)=sum(vinstvec(k:end))/length(vinstvec(k:end));
```

```
        topp3procentvec(k)=sum(topp3vec(k:end))/length(
          topp3vec(k:end));

        if sum(avec(k:end)) >0
            vinstprocentavec(k)=sum(vinstavec(k:end))/sum(
              avec(k:end));
            topp3procentavec(k)=sum(topp3avec(k:end))/sum(
              avec(k:end));
        end

        if sum(vvec(k:end)) >0
            vinstprocentvvec(k)=sum(vinstvvec(k:end))/sum(
              vvec(k:end));
            topp3procentvvec(k)=sum(topp3vvec(k:end))/sum(
              vvec(k:end));
        end
    end

    for k = 1:length(redhorse.resultat)-5
        n = k+1;

        if avec(k)==1 && vinstprocentavec(n)/
          vinstprocentvec(n) > 1.05
            gvek(autotrivselindex) = 1;
        end

        if avec(k)==1 && topp3procentavec(n)/
          topp3procentvec(n) > 1.05
            hvek(autotrivselindex) = 1;
        end

        if vvec(k)==1 && vinstprocentvvec(n)/
          vinstprocentvec(n) > 1.05
            gvek(autotrivselindex) = 1;
        end

        if vvec(k)==1 && topp3procentvvec(n)/
          topp3procentvec(n) > 1.05
            hvek(autotrivselindex) = 1;
        end
        autotrivselindex=autotrivselindex+1;
    end

    for k = 1:5
        gvek(autotrivselindex) = 0;
        hvek(autotrivselindex) = 0;
        autotrivselindex = autotrivselindex + 1;
    end
else
```

```
    for k = 1:length(redhorse.resultat)
    gvek(autotrivselindex) = 0;
    hvek(autotrivselindex) = 0;
    autotrivselindex = autotrivselindex + 1;
end
end
```

```
function [evek, fvek, distanstrivselindex] = trivseldistans(evek,fvek, distanstrivselindex, redhorse, topp3, won, topp3procent, vinstprocent)

if length(redhorse.resultat) > 5
    vinstvec = zeros(length(redhorse.resultat),1);
    topp3vec = zeros(length(redhorse.resultat),1);
    kdistvec = zeros(length(redhorse.resultat),1);
    mdistvec = zeros(length(redhorse.resultat),1);
    ldistvec = zeros(length(redhorse.resultat),1);

    resultat = redhorse.resultat;

    vinstprocentvec = zeros(length(redhorse.resultat),1);
    topp3procentvec = zeros(length(redhorse.resultat),1);
    vinstprocentkdistvec=zeros(length(redhorse.resultat),1);
    topp3procentkdistvec=zeros(length(redhorse.resultat),1);
    vinstprocentmdistvec=zeros(length(redhorse.resultat),1);
    topp3procentmdistvec=zeros(length(redhorse.resultat),1);
    vinstprocentldistvec=zeros(length(redhorse.resultat),1);
    topp3procentldistvec=zeros(length(redhorse.resultat),1);

    for k = 1:length(resultat)
        if resultat(k)==1
            vinstvec(k)=1;
        end
        if resultat(k) < 4 && resultat(k) > 0
            topp3vec(k)=1;
        end
        dist=redhorse.distans(k);

        if isnan(dist)==1
        elseif dist <1700
            kdistvec(k) = 1;
        elseif dist > 2400
            ldistvec(k) = 1;
```

```
        else
            mdistvec(k) = 1;
        end
    end

    vinstkdistvec=vinstvec.*kdistvec;
    vinstmdistvec=vinstvec.*mdistvec;
    vinstldistvec=vinstvec.*ldistvec;
    topp3kdistvec=topp3vec.*kdistvec;
    topp3mdistvec=topp3vec.*mdistvec;
    topp3ldistvec=topp3vec.*ldistvec;

    for k = 1:length(redhorse.resultat)
        vinstprocentvec(k)=sum(vinstvec(k:end))/length(vinstvec(k:end));
        topp3procentvec(k)=sum(topp3vec(k:end))/length(topp3vec(k:end));

        if sum(kdistvec(k:end)) >0
            vinstprocentkdistvec(k)=sum(vinstkdistvec(k:end))/sum(kdistvec(k:end));
            topp3procentkdistvec(k)=sum(topp3kdistvec(k:end))/sum(kdistvec(k:end));
        end

        if sum(mdistvec(k:end)) >0
            vinstprocentmdistvec(k)=sum(vinstmdistvec(k:end))/sum(mdistvec(k:end));
            topp3procentmdistvec(k)=sum(topp3mdistvec(k:end))/sum(mdistvec(k:end));
        end

        if sum(ldistvec(k:end)) >0
            vinstprocentldistvec(k)=sum(vinstldistvec(k:end))/sum(ldistvec(k:end));
            topp3procentldistvec(k)=sum(topp3ldistvec(k:end))/sum(ldistvec(k:end));
        end
    end

    for k = 1:length(redhorse.resultat)-5
        n = k+1;
        if kdistvec(k)==1 && vinstprocentkdistvec(n)/vinstprocentvec(n) > 1.05
            evek(distanstrivselindex) = 1;
        end

        if kdistvec(k)==1 && topp3procentkdistvec(n)/topp3procentvec(n) > 1.05
```

```
            fvek(distanstrivselindex) = 1;
        end

        if mdistvec(k)==1 && vinstprocentmdistvec(n)/
          vinstprocentvec(n) > 1.05
            evek(distanstrivselindex) = 1;
        end

        if mdistvec(k)==1 && topp3procentmdistvec(n)/
          topp3procentvec(n) > 1.05
            fvek(distanstrivselindex) = 1;
        end

        if ldistvec(k)==1 && vinstprocentldistvec(n)/
          vinstprocentvec(n) > 1.05
            evek(distanstrivselindex) = 1;
        end

        if ldistvec(k)==1 && topp3procentldistvec(n)/
          topp3procentvec(n) > 1.05
            fvek(distanstrivselindex) = 1;
        end
        distanstrivselindex=distanstrivselindex+1;
    end

    for k = 1:5
        evek(distanstrivselindex) = 0;
        fvek(distanstrivselindex) = 0;
        distanstrivselindex = distanstrivselindex + 1;
    end
else
    for k = 1:length(redhorse.resultat)
        evek(distanstrivselindex) = 0;
        fvek(distanstrivselindex) = 0;
        distanstrivselindex = distanstrivselindex + 1;
    end
end
```

```
function [sommarvinsttrivsel, sommartopp3trivsel,
  sommartrivselindex] = trivselsommar(sommarvinsttrivsel,
  sommartopp3trivsel, sommartrivselindex, redhorse, topp3,
  won, topp3procent, vinstprocent)

if length(redhorse.resultat) > 5
    vinstvec = zeros(length(redhorse.resultat),1);
    topp3vec = zeros(length(redhorse.resultat),1);
    sommarvec = zeros(length(redhorse.resultat),1);
    vintervec = zeros(length(redhorse.resultat),1);
    resultat = redhorse.resultat;
```

```
    vinstprocentvec = zeros(length(redhorse.resultat),1);
    topp3procentvec = zeros(length(redhorse.resultat),1);
    vinstprocentsommarvec=zeros(length(redhorse.resultat)
      ,1);
    topp3procentsommarvec=zeros(length(redhorse.resultat)
      ,1);
    vinstprocentvintervec=zeros(length(redhorse.resultat)
      ,1);
    topp3procentvintervec=zeros(length(redhorse.resultat)
      ,1);

    for k = 1:length(resultat)
        if resultat(k)==1
            vinstvec(k)=1;
        end
        if resultat(k) < 4 && resultat(k) > 0
            topp3vec(k)=1;
        end

        a = redhorse.datum(k);
        b = floor(a/100);
        c = floor(b/100)*100;
        month = b-c;

        if month >= 4 && month <= 9
            sommarvec(k) = 1;
        else
            vintervec(k) = 1;
        end
    end

    vinstvintervec=vinstvec.*vintervec;
    vinstsommarvec=vinstvec.*sommarvec;
    topp3vintervec=topp3vec.*vintervec;
    topp3sommarvec=topp3vec.*sommarvec;

    for k = 1:length(redhorse.resultat)
        vinstprocentvec(k)=sum(vinstvec(k:end))/length(
          vinstvec(k:end));
        topp3procentvec(k)=sum(topp3vec(k:end))/length(
          topp3vec(k:end));

        if sum(sommarvec(k:end)) >0
            vinstprocentsommarvec(k)=sum(vinstsommarvec(k:
              end))/sum(sommarvec(k:end));
            topp3procentsommarvec(k)=sum(topp3sommarvec(k:
              end))/sum(sommarvec(k:end));
        end

```

```
        if sum(vintervec(k:end)) >0
            vinstprocentvintervec(k)=sum(vinstvintervec(k:end))/sum(vintervec(k:end));
            topp3procentvintervec(k)=sum(topp3vintervec(k:end))/sum(vintervec(k:end));
        end
    end

    for k = 1:length(redhorse.resultat)-5
        n = k+1;

        if sommarvec(k)==1 && vinstprocentsommarvec(n)/vinstprocentvec(n) > 1.05
            sommarvinsttrivsel(sommartrivselindex) = 1;
        end

        if sommarvec(k)==1 && topp3procentsommarvec(n)/topp3procentvec(n) > 1.05
            sommartopp3trivsel(sommartrivselindex) = 1;
        end

        if vintervec(k)==1 && vinstprocentvintervec(n)/vinstprocentvec(n) > 1.05
            sommarvinsttrivsel(sommartrivselindex) = 1;
        end

        if vintervec(k)==1 && topp3procentvintervec(n)/topp3procentvec(n) > 1.05
            sommartopp3trivsel(sommartrivselindex) = 1;
        end
        sommartrivselindex=sommartrivselindex+1;
    end

    for k = 1:5
        sommarvinsttrivsel(sommartrivselindex) = 0;
        sommartopp3trivsel(sommartrivselindex) = 0;
        sommartrivselindex = sommartrivselindex + 1;
    end
else
    for k = 1:length(redhorse.resultat)
        sommarvinsttrivsel(sommartrivselindex) = 0;
        sommartopp3trivsel(sommartrivselindex) = 0;
        sommartrivselindex = sommartrivselindex + 1;
    end
end
```

```
%% The data set for model 2 and model 3. I.e. only with whole races.

```

```
load superhorse % Load data set for model 1

load okloppid % Load id vector with whole races

minisuperhorse = struct;minisuperhorse.namn = []; minisuperhorse.id = [];minisuperhorse.color = [];
minisuperhorse.gender = [];minisuperhorse.birthdate = [];
minisuperhorse.ras = [];minisuperhorse.loppid = []; minisuperhorse.datum = [];
minisuperhorse.startnummer = [];minisuperhorse.loppnummer = [];minisuperhorse.distans=[];
minisuperhorse.tid = [];minisuperhorse.resultat=[]; minisuperhorse.ntungbana = [];
minisuperhorse.odds = [];minisuperhorse.kusk = {};
minisuperhorse.trainer = [];minisuperhorse.vinterbana = [];
minisuperhorse.tungbana = [];minisuperhorse.autostart = []; minisuperhorse.galopp = [];
minisuperhorse.skorfram = [];minisuperhorse.skorbak = [];
minisuperhorse.vinst = []; minisuperhorse.bana = [];
minisuperhorse.distansdummy = []; minisuperhorse.pengar = []; minisuperhorse.alder = [];
minisuperhorse.trivselfaktor = []; minisuperhorse.brakusk = []; minisuperhorse.mycketbrakusk = [];
minisuperhorse.ganskabrakusk = []; minisuperhorse.kuskrank = []; minisuperhorse.valack = [];
minisuperhorse.sto = []; minisuperhorse.hingst = []; minisuperhorse.mktbrahorse = [];
minisuperhorse.ganskabrahorse = []; minisuperhorse.brahorse = []; minisuperhorse.vinstprocent = [];
minisuperhorse.platsprocent = []; minisuperhorse.startp = []; minisuperhorse.vinstform = [];
minisuperhorse.segerform = [];minisuperhorse.banavinsttrivsel=[];minisuperhorse.banatopp3trivsel=[];
minisuperhorse.distansvinsttrivsel=[];minisuperhorse.distanstopp3trivsel=[];
minisuperhorse.autovinsttrivsel=[];minisuperhorse.autotopp3trivsel=[];
minisuperhorse.sommarvinsttrivsel=[];minisuperhorse.sommartopp3trivsel=[];
minisuperhorse.mintid = [];

for j=okloppid'
    redhorse = struct;
    redhorse.namn = [];
    redhorse.id = [];
    redhorse.color = [];
    redhorse.gender = [];
    redhorse.birthdate = [];redhorse.ras = [];redhorse.loppid = [];redhorse.datum = [];
```

```
    redhorse.startnummer = [];redhorse.loppnummer = [];
      redhorse.distans=[];
    redhorse.tid = [];redhorse.resultat=[];redhorse.
      ntungbana = [];redhorse.odds = [];
    redhorse.kusk = [];redhorse.trainer = [];redhorse.
      vinterbana = [];
    redhorse.tungbana = [];redhorse.autostart = [];redhorse
      .galopp = [];
    redhorse.skorfram = [];redhorse.skorbak = [];redhorse.
      vinst = []; redhorse.bana = [];
    redhorse.distansdummy = [];

    indexvec = find(superhorse.loppid==j);
    redhorse.namn = superhorse.namn(indexvec);
    redhorse.loppid = superhorse.loppid(indexvec);
    redhorse.id = superhorse.id(indexvec);
    redhorse.resultat = superhorse.resultat(indexvec);
    redhorse.startnummer = superhorse.startnummer(indexvec)
      ;
    redhorse.loppnummer = superhorse.loppnummer(indexvec);
    redhorse.distans = superhorse.distans(indexvec);
    redhorse.tid = superhorse.tid(indexvec);
    redhorse.ntungbana = superhorse.ntungbana(indexvec);
    redhorse.vinterbana = superhorse.vinterbana(indexvec);
    redhorse.tungbana = superhorse.tungbana(indexvec);
    redhorse.odds = superhorse.odds(indexvec);
    redhorse.kusk = superhorse.kusk(indexvec);
    redhorse.trainer = superhorse.trainer(indexvec);
    redhorse.autostart = superhorse.autostart(indexvec);
    redhorse.galopp = superhorse.galopp(indexvec);
    redhorse.skorfram = superhorse.skorfram(indexvec);
    redhorse.skorbak = superhorse.skorbak(indexvec);
    redhorse.vinst = superhorse.vinst(indexvec);
    redhorse.alder = superhorse.alder(indexvec);
    redhorse.vinstform = superhorse.vinstform(indexvec);
    redhorse.startp = superhorse.startp(indexvec);
    redhorse.datum = superhorse.datum(indexvec);
    redhorse.trivselfaktor = superhorse.trivselfaktor(
      indexvec);
    redhorse.brakusk = superhorse.brakusk(indexvec);
    redhorse.ganskabrakusk = superhorse.ganskabrakusk(
      indexvec);
    redhorse.kuskrank = superhorse.kuskrank(indexvec);
    redhorse.valack = superhorse.valack(indexvec);
    redhorse.hingst = superhorse.hingst(indexvec);
    redhorse.sto = superhorse.sto(indexvec);
    redhorse.mycketbrakusk = superhorse.mycketbrakusk(
      indexvec);
```

```
    redhorse.mktbrahorse = superhorse.mktbrahorse(indexvec);
    redhorse.brahorse = superhorse.brahorse(indexvec);
    redhorse.ganskabrahorse = superhorse.ganskabrahorse(indexvec);
    redhorse.vinstprocent = superhorse.vinstprocent(indexvec);
    redhorse.pengar = superhorse.pengar(indexvec);
    redhorse.platsprocent = superhorse.platsprocent(indexvec);
    redhorse.segerform = superhorse.segerform(indexvec);
    redhorse.mintid = superhorse.mintid(indexvec);
    redhorse.banavinsttrivsel=superhorse.banavinsttrivsel(indexvec);
    redhorse.banatopp3trivsel=superhorse.banatopp3trivsel(indexvec);
    redhorse.distansvinsttrivsel=superhorse.distansvinsttrivsel(indexvec);
    redhorse.distanstopp3trivsel=superhorse.distanstopp3trivsel(indexvec);
    redhorse.autovinsttrivsel=superhorse.autovinsttrivsel(indexvec);
    redhorse.autotopp3trivsel=superhorse.autotopp3trivsel(indexvec);
    redhorse.sommarvinsttrivsel=superhorse.sommarvinsttrivsel(indexvec);
    redhorse.sommartopp3trivsel=superhorse.sommartopp3trivsel(indexvec);

    mintid2=[];
        for k=1:length(redhorse.resultat)
            if redhorse.mintid(k) < 9
                redhorse.mintid(k) = 0;
            elseif redhorse.mintid(k) >=9
                mintid2=[mintid2;redhorse.mintid(k)];
            end
        end
    mintid1=min(mintid2);
    maxtid1=max(mintid2);
    medeltid=(mintid1+maxtid1)/2;

    if isempty(medeltid)==1
        medeltid=1;
        mintid1=1;
    end

    for k=1:length(redhorse.resultat)
        if redhorse.mintid(k) == 0
            redhorse.mintid(k) = medeltid;
```

```
        end
    end

    maxstartp = max(redhorse.startp);
    maxpengar = max(redhorse.pengar);
    maxvinstform = max(redhorse.vinstform);
    maxtrivsel = max(redhorse.trivselfaktor);
    distanser = unique(redhorse.distans);
    distansdummy = zeros(length(redhorse.distans),1);

    if length(distanser) > 1
        mindistans = min(distanser);
        index = find(redhorse.distans>mindistans);
        distansdummy(index)=1;
    end

    minisuperhorse.loppid = [minisuperhorse.loppid;
      redhorse.loppid];
    minisuperhorse.id = [minisuperhorse.id ; redhorse.id];
    minisuperhorse.resultat = [minisuperhorse.resultat ;
      redhorse.resultat];
    minisuperhorse.startnummer = [minisuperhorse.
      startnummer ; redhorse.startnummer];
    minisuperhorse.loppnummer = [minisuperhorse.loppnummer
      ; redhorse.loppnummer];
    minisuperhorse.distans = [minisuperhorse.distans ;
      redhorse.distans];
    minisuperhorse.tid = [minisuperhorse.tid ; redhorse.tid
      ];
    minisuperhorse.ntungbana = [minisuperhorse.ntungbana ;
      redhorse.ntungbana];
    minisuperhorse.vinterbana = [minisuperhorse.vinterbana
      ; redhorse.vinterbana];
    minisuperhorse.tungbana = [minisuperhorse.tungbana ;
      redhorse.tungbana];
    minisuperhorse.odds = [minisuperhorse.odds ; redhorse.
      odds];
    minisuperhorse.kusk = [minisuperhorse.kusk redhorse.
      kusk];
    minisuperhorse.trainer = [minisuperhorse.trainer
      redhorse.trainer];
    minisuperhorse.autostart = [minisuperhorse.autostart ;
      redhorse.autostart];
    minisuperhorse.galopp = [minisuperhorse.galopp ;
      redhorse.galopp];
    minisuperhorse.skorfram = [minisuperhorse.skorfram ;
      redhorse.skorfram];
    minisuperhorse.skorbak = [minisuperhorse.skorbak ;
      redhorse.skorbak];
```

```
minisuperhorse.vinst = [minisuperhorse.vinst ; redhorse
  .vinst];
minisuperhorse.alder = [minisuperhorse.alder ; redhorse
  .alder];
minisuperhorse.datum = [minisuperhorse.datum ; redhorse
  .datum];
minisuperhorse.namn = [minisuperhorse. ; redhorse.namn
  ];
minisuperhorse.brakusk = [minisuperhorse.brakusk ;
  redhorse.brakusk];
minisuperhorse.ganskabrakusk = [minisuperhorse.
  ganskabrakusk ; redhorse.ganskabrakusk];
minisuperhorse.kuskrank = [minisuperhorse.kuskrank ;
  redhorse.kuskrank];
minisuperhorse.valack = [minisuperhorse.valack ;
  redhorse.valack];
minisuperhorse.hingst = [minisuperhorse.hingst ;
  redhorse.hingst];
minisuperhorse.sto = [minisuperhorse.sto ; redhorse.sto
  ];
minisuperhorse.mycketbrakusk = [minisuperhorse.
  mycketbrakusk ; redhorse.mycketbrakusk];
minisuperhorse.mktbrahorse = [minisuperhorse.
  mktbrahorse ; redhorse.mktbrahorse];
minisuperhorse.brahorse = [minisuperhorse.brahorse ;
  redhorse.brahorse];
minisuperhorse.ganskabrahorse = [minisuperhorse.
  ganskabrahorse ; redhorse.ganskabrahorse];
minisuperhorse.vinstprocent = [minisuperhorse.
  vinstprocent ; redhorse.vinstprocent];
minisuperhorse.startp = [minisuperhorse.startp ;
  redhorse.startp/maxstartp];
minisuperhorse.pengar = [minisuperhorse.pengar;
  redhorse.pengar/maxpengar];
minisuperhorse.vinstform = [minisuperhorse.vinstform;
  redhorse.vinstform/maxvinstform];
minisuperhorse.distansdummy = [minisuperhorse.
  distansdummy; distansdummy];
minisuperhorse.platsprocent = [minisuperhorse.
  platsprocent; redhorse.platsprocent];
minisuperhorse.trivselfaktor = [minisuperhorse.
  trivselfaktor; redhorse.trivselfaktor/maxtrivsel];
minisuperhorse.segerform = [minisuperhorse.segerform
  redhorse.segerform];
%minisuperhorse.cvek = [minisuperhorse.cvek; redhorse.
  cvek];
minisuperhorse.mintid = [minisuperhorse.mintid;
  redhorse.mintid/mintid1];
```

```
    minisuperhorse.banavinsttrivsel=[minisuperhorse.banavinsttrivsel;redhorse.banavinsttrivsel];
    minisuperhorse.banatopp3trivsel=[minisuperhorse.banatopp3trivsel;redhorse.banatopp3trivsel];
    minisuperhorse.distansvinsttrivsel=[minisuperhorse.distansvinsttrivsel;redhorse.distansvinsttrivsel];
    minisuperhorse.distanstopp3trivsel=[minisuperhorse.distanstopp3trivsel;redhorse.distanstopp3trivsel];
    minisuperhorse.autovinsttrivsel=[minisuperhorse.autovinsttrivsel;redhorse.autovinsttrivsel];
    minisuperhorse.autotopp3trivsel=[minisuperhorse.autotopp3trivsel;redhorse.autotopp3trivsel];
    minisuperhorse.sommarvinsttrivsel=[minisuperhorse.sommarvinsttrivsel;redhorse.sommarvinsttrivsel];
    minisuperhorse.sommartopp3trivsel=[minisuperhorse.sommartopp3trivsel;redhorse.sommartopp3trivsel];
end
```

Bilaga C

MATLAB Scripts - Models

```
%% Model 2: Find best covariates based on AIC
tic

load('minisuperhorse10') % Loading data
load('predictionhorse10')

Y =log(minisuperhorse.omvresultat');

minAICvec = [];minRMSvec = [];minAICvecPRED = []; minRMSvecPRED = [];

covariates=[{'odds'} {'pengar'} {'alder' 'brakusk'} {'mycketbrakusk'} {'ganskabrakusk'} {'mktbrahorse'}...
{'ganskabrahorse'} {'brahorse'} {'sto'} {'hingst'} {'vinstprocent'} {'platsprocent'}...
{'startp'} {'vinstform'} {'segerform'} {'mintid'} {'StartnummerRank'}...
{'distansvinsttrivsel'} {'distanstopp3trivsel'} {'sommarvinsttrivsel'} {'sommartopp3trivsel'}...
{'autovinsttrivsel'} {'autotopp3trivsel'} {'distansdummy'} {'autostart'}]';

covatiatematrixReg = [minisuperhorse.odds minisuperhorse.pengar minisuperhorse.alder...
minisuperhorse.brakusk minisuperhorse.mycketbrakusk minisuperhorse.ganskabrakusk...
minisuperhorse.mktbrahorse minisuperhorse.ganskabrahorse minisuperhorse.brahorse...
minisuperhorse.sto minisuperhorse.hingst minisuperhorse.vinstprocent...
minisuperhorse.platsprocent minisuperhorse.startp minisuperhorse.vinstform...
```

```
minisuperhorse.segerform' minisuperhorse.mintid
  minisuperhorse.StartnummerRank...
minisuperhorse.distansvinsttrivsel minisuperhorse.
  distanstopp3trivsel...
minisuperhorse.sommarvinsttrivsel minisuperhorse.
  sommartopp3trivsel...
minisuperhorse.autovinsttrivsel minisuperhorse.
  autotopp3trivsel...
minisuperhorse.distansdummy minisuperhorse.autostart];

Xreg = [];squared = [];sqroot = [];squaredcov = [];
  sqrootcov = [];multcov = [];alonecov = [];
antcov = length(covatiatematrixReg(1,:));ant = 1;vek = [];

for k = 1:antcov
    alonecov = [alonecov; covariates(k) {'alone'}];
    squared = [squared covatiatematrixReg(:,k).^2];
    squaredcov = [squaredcov; covariates(k) {'squared'}];
    sqroot = [sqroot covatiatematrixReg(:,k).^1/2];
    sqrootcov = [sqrootcov; covariates(k) {'root'}];
    for n = k+1:antcov
        if sum(covatiatematrixReg(:,k).*covatiatematrixReg
          (:,n)) ~=0
            Xreg=[Xreg covatiatematrixReg(:,k).*
              covatiatematrixReg(:,n)];
            multcov = [multcov; covariates(k) covariates(n)
              ];
        end
    end
end

finalcovariates = [alonecov; squaredcov ;sqrootcov; multcov
  ;];
covatiatematrixReg = [covatiatematrixReg sqroot squared
  Xreg];

covatiatematrixPred = [predictionhorse.odds predictionhorse
  .pengar predictionhorse.alder...
predictionhorse.brakusk predictionhorse.mycketbrakusk
  predictionhorse.ganskabrakusk...
predictionhorse.mktbrahorse predictionhorse.ganskabrahorse
  predictionhorse.brahorse...
predictionhorse.sto predictionhorse.hingst predictionhorse.
  vinstprocent...
predictionhorse.platsprocent predictionhorse.startp
  predictionhorse.vinstform...
predictionhorse.segerform predictionhorse.mintid
  predictionhorse.StartnummerRank...
```

```
predictionhorse.distansvinsttrivsel predictionhorse.
  distanstopp3trivsel...
predictionhorse.sommarvinsttrivsel predictionhorse.
  sommartopp3trivsel...
predictionhorse.autovinsttrivsel predictionhorse.
  autotopp3trivsel...
predictionhorse.distansdummy predictionhorse.autostart];

Xpred=[];squared = [];sqroot1 = [];

for t = 1:antcov
    squared = [squared covatiatematrixPred(:,t).^2];
    sqroot1 = [sqroot1 covatiatematrixPred(:,t).^1/2];
    for n = t+1:antcov
        if sum(covatiatematrixReg(:,t).*covatiatematrixReg
          (:,n)) ~=0
            Xpred=[Xpred covatiatematrixPred(:,t).*
              covatiatematrixPred(:,n)];
        end
    end
end

covatiatematrixPred = [covatiatematrixPred sqroot1 squared
  Xpred];

AICvec=[];rmsvec=[];[rader kolonner] = size(
  covatiatematrixReg);XregFINAL = [];
XpredFINAL = [];used =[];okindex = 1:kolonner;n=1;

for k = 1:300
    AICvec=[];
    rmsvec=[];
    for m =okindex
        Xreg = XregFINAL;
        antkol=m;
        Xreg=[Xreg covatiatematrixReg(:,m)];
        X=[ones(length(Y),1) Xreg];

        b = regress(Y,X);
        [AICvec(m) SSE rmsvec(m)] = modellanalys(b, log(
          minisuperhorse.omvresultat'), X);

        if sum(find(b==0))>=1
            AICvec(m)=10e20;
        else
            [AICvec(m) SSE rmsvec(m)] = modellanalys(b, log
              (minisuperhorse.omvresultat'), X);
        end
    n=n+1;
```

```
end
AICvec(used)=10e20;
a=find(AICvec == min(AICvec));
used =[used;a(1)];
XregFINAL = [XregFINAL covatiatematrixReg(:,a(1))];
XpredFINAL = [XpredFINAL covatiatematrixPred(:,a(1))];
okindex(find(okindex==a(1)))=[];
minAICvec = [minAICvec; min(AICvec)];
minRMSvec = [minRMSvec; rmsvec(a(1))];
X=[ones(length(Y),1) XregFINAL];
b = regress(Y,X);
[AIC SSE rms] = modellanalys(b, Y, X);
minAICvecPRED = [minAICvecPRED; AIC];
minRMSvecPRED = [minRMSvecPRED; rms];
[minAICvec minAICvecPRED]
end
toc
```

```
%% Modell 2

load minisuperhorse10INKLloppid % Loading data
load predictionhorse10

covariates=[{'odds'} {'pengar'} {'alder' 'brakusk'} {'mycketbrakusk'} {'ganskabrakusk'} {'mktbrahorse'}...
{'ganskabrahorse'} {'brahorse'} {'sto'} {'hingst'} {'vinstprocent'} {'platsprocent'}...
{'startp'} {'vinstform'} {'segerform'} {'mintid'} {'StartnummerRank'}...
{'distansvinsttrivsel'} {'distanstopp3trivsel'} {'sommarvinsttrivsel'} {'sommartopp3trivsel'}...
{'autovinsttrivsel'} {'autotopp3trivsel'} {'distansdummy'} {'autostart'}]';

covariatematrixReg = [minisuperhorse.odds minisuperhorse.pengar minisuperhorse.alder...
minisuperhorse.brakusk minisuperhorse.mycketbrakusk minisuperhorse.ganskabrakusk...
minisuperhorse.mktbrahorse minisuperhorse.ganskabrahorse minisuperhorse.brahorse...
minisuperhorse.sto minisuperhorse.hingst minisuperhorse.vinstprocent...
minisuperhorse.platsprocent minisuperhorse.startp minisuperhorse.vinstform...
minisuperhorse.segerform' minisuperhorse.mintid minisuperhorse.StartnummerRank...
minisuperhorse.distansvinsttrivsel minisuperhorse.distanstopp3trivsel...
```

```
minisuperhorse.sommarvinsttrivsel minisuperhorse.sommartopp3trivsel...
minisuperhorse.autovinsttrivsel minisuperhorse.autotopp3trivsel...
minisuperhorse.distansdummy minisuperhorse.autostart];

covariatematrixPred = [predictionhorse.odds predictionhorse.pengar predictionhorse.alder...
predictionhorse.brakusk predictionhorse.mycketbrakusk predictionhorse.ganskabrakusk...
predictionhorse.mktbrahorse predictionhorse.ganskabrahorse predictionhorse.brahorse...
predictionhorse.sto predictionhorse.hingst predictionhorse.vinstprocent...
predictionhorse.platsprocent predictionhorse.startp predictionhorse.vinstform...
predictionhorse.segerform predictionhorse.mintid predictionhorse.StartnummerRank...
predictionhorse.distansvinsttrivsel predictionhorse.distanstopp3trivsel...
predictionhorse.sommarvinsttrivsel predictionhorse.sommartopp3trivsel...
predictionhorse.autovinsttrivsel predictionhorse.autotopp3trivsel...
predictionhorse.distansdummy predictionhorse.autostart];

squared = [];sqroot = [];squaredcov = [];sqrootcov = [];multcov = [];alonecov = [];
antcov = length(covariatematrixReg(1,:));RMSdata=[];ant = 1;vek = [];Xreg=[];

for k = 1:antcov
    alonecov = [alonecov; covariates(k) {'alone'}];
    squared = [squared covariatematrixReg(:,k).^2];
    squaredcov = [squaredcov; covariates(k) {'squared'}];
    sqroot = [sqroot covariatematrixReg(:,k).^1/2];
    sqrootcov = [sqrootcov; covariates(k) {'root'}];
    for n = k+1:antcov
        if sum(covariatematrixReg(:,k).*covariatematrixReg(:,n)) ~=0
            Xreg=[Xreg covariatematrixReg(:,k).*covariatematrixReg(:,n)];
            multcov = [multcov; covariates(k) covariates(n)];
        end
    end
end

```

```
finalcovariates = [alonecov; squaredcov ;sqrootcov; multcov
  ;];
covariatematrixReg = [covariatematrixReg sqroot squared
  Xreg];

Xpred=[];squared = [];sqroot1 = [];

for t = 1:antcov
    squared = [squared covariatematrixPred(:,t).^2];
    sqroot1 = [sqroot1 covariatematrixPred(:,t).^1/2];
    for n = t+1:antcov
        if sum(covariatematrixReg(:,t).*covariatematrixReg
          (:,n)) ~=0
            Xpred=[Xpred covariatematrixPred(:,t).*
              covariatematrixPred(:,n)];
        end
    end
end
covariatematrixPred = [covariatematrixPred sqroot1 squared
  Xpred];

% Dependent variable
Y = log(minisuperhorse.omvresultat');
Ylogistisk = zeros(length(Y),1);
for i = 1:length(Y)
    if minisuperhorse.omvresultat(i)==1
        Ylogistisk(i)=1;
    end
end

load used
used=used(1:35); % Covariates
Xreg=covariatematrixReg(:,used);

% Regression
beta = Xreg\Y;

% Predict
Xpred=covariatematrixPred(:,used);
prediktion = exp(Xpred*beta);
pred2 = prediktion;
[dagskassa antalhorses jmntmatris SOS SOSo] = analys(
  predictionhorse,prediktion);
plot(1:length(dagskassa),dagskassa,1:length(dagskassa),
  dagskassa,'r.')

% Calculate RMSE for the model
unika=unique(minisuperhorse.loppid);
p=[];
```

```
Xreg=covariatematrixReg(:,used);
beta = Xreg\Y;
prediktion=exp(Xreg*beta);

for i=unika'
    plats=find(minisuperhorse.loppid==i);
    predtemp=prediktion(plats);
    ptemp = exp2prob(predtemp');
    p=[p;ptemp(:,1)];
end

SSE=sum((Ylogistisk-p).^2);
RMSE=sqrt(SSE/length(Ylogistisk));

SSE=sum((Ylogistisk-prediktiondataNormerad).^2);
RMSElogistisk=sqrt(SSE/length(Ylogistisk));

%% Predict seven V75 days
datvektor=[130413 130406 130223 130105 121201 121020 121013 ];
idvektor=[529859 530004 529394 529398 521831 521823 522118 ];
SOS=0;
SOSo=0;

load Predictionhorse %Data

dagskassa=1000;
for i=1:7
    predictionhorse=Predictionhorse(i);
    covariatematrixPred = [predictionhorse.odds predictionhorse.pengar predictionhorse.alder...
    predictionhorse.brakusk predictionhorse.mycketbrakusk predictionhorse.ganskabrakusk...
    predictionhorse.mktbrahorse predictionhorse.ganskabrahorse predictionhorse.brahorse...
    predictionhorse.sto predictionhorse.hingst predictionhorse.vinstprocent...
    predictionhorse.platsprocent predictionhorse.startp predictionhorse.vinstform...
    predictionhorse.segerform predictionhorse.mintid predictionhorse.StartnummerRank...
    predictionhorse.distansvinsttrivsel predictionhorse.distanstopp3trivsel...
    predictionhorse.sommarvinsttrivsel predictionhorse.sommartopp3trivsel...
    predictionhorse.autovinsttrivsel predictionhorse.autotopp3trivsel...
```

```
    predictionhorse.distansdummy predictionhorse.autostart ];
    squared = [];sqroot = [];squaredcov = [];sqrootcov = [];multcov = [];alonecov = [];
    antcov = length(covariatematrixPred(1,:));RMSdata=[]; ant = 1;vek = [];Xpred=[];

        for k = 1:antcov
            squared = [squared covariatematrixPred(:,k).^2];
            sqroot = [sqroot covariatematrixPred(:,k).^1/2];
            for n = k+1:antcov
                if sum(covariatematrixReg(:,k).*covariatematrixReg(:,n)) ~=0
                    Xpred=[Xpred covariatematrixPred(:,k).*covariatematrixPred(:,n)];
                end
            end
        end

    finalcovariates = [alonecov; squaredcov ;sqrootcov; multcov;];
    covariatematrixPred = [covariatematrixPred sqroot squared Xpred];
    Xpred=covariatematrixPred(:,used);
    prediktion = exp(Xpred*beta);
    [dagskassa SOSp SOSo] = analysmedopt(dagskassa, predictionhorse,prediktion);
    dagskassa
end
```

```
function [dagskassa antalhorses jmntmatris SOS SOSo]= analys(predictionhorse, prediktion)
SOS = 0;
SOSo = 0;
SOS3 = 0;
SOSo3 = 0;
SOS1 = 0;
SOSo1 = 0;

% Real result
resultat = predictionhorse.omvresultat';
odds=predictionhorse.odds;

unkiloppid=unique(predictionhorse.loppid);
idmatris = zeros(15,7);
idrankodds = zeros(15,7);
idrankpred = zeros(15,7);
```

```
predrankvinnare = [];
oddsrankvinnare = [];
antalhorses = [];
kassa=1000;
dagskassa=[kassa];
sannolikhet=[];

for j=1:length(unkiloppid)
    plats=find(predictionhorse.loppid==unkiloppid(j));
    rank=tiedrank(prediktion(plats));
    prediktiontemp = prediktion(plats);
    tiedrankodds=tiedrank(odds(plats));
    odds1=odds(plats)/10;
    prediktion1=prediktion(plats);
    resultat1=resultat(plats);
    prob = exp2prob(prediktiontemp');
    prob = prob(:,1);
    odds2 = 1./odds1;
    probodds = odds2./(sum(odds2));
    sannolikhet=[sannolikhet; prob];

    m=[resultat1' tiedrankodds rank odds1 prediktiontemp/
      sum(prediktiontemp)*sum(1:length(rank))    prob(:,1)
      probodds prob(:,1)-odds2];
    [B I]=sort(m(:,1));
    m2=m(I,:);
    disp(m2)

    asd=find((prob-odds2)>0);
    betprob=max(prob(asd));
    if isempty(betprob)==0
        k=find(betprob==prob);
        oddsbet=odds1(k);
        if betprob > 0.2
            summa = 200;
        else
            summa=100;
        end
        if betprob>0.1
            placering=resultat1(k);
            if placering==1
                kassa=kassa+summa*(odds1(k)-1);
            else
                kassa=kassa-summa;
            end
        end
    end
dagskassa=[dagskassa; kassa];
end
```

```
function [antalhorses jmntmatris SOS SOSo]= analysmedopt(
  predictionhorse, prediktion)
% Real result
resultat = predictionhorse.omvresultat';
odds=predictionhorse.odds;

unkiloppid=unique(predictionhorse.loppid);

idmatris = zeros(15,7);idrankodds = zeros(15,7);idrankpred
  = zeros(15,7);predrankvinnare = [];
oddsrankvinnare = [];antalhorses = [];jmnt = [];jmnt2 = [];
  jmnt3 = [];jmnt4 = [];jmnt5 = [];
jmnt6 = [];jmnt7 = [];jmntmatris = [];probmatris=[];
  probunsorted=[];
proboddsmatris=[];oddsmatris=[];sannolikhet=[];

for j=1:length(unkiloppid)
    plats=find(predictionhorse.loppid==unkiloppid(j));
    rank=tiedrank(prediktion(plats));
    prediktiontemp = prediktion(plats);
    tiedrankodds=tiedrank(odds(plats));
    odds1=odds(plats)/10;
    prediktion1=prediktion(plats);
    resultat1=resultat(plats);
    prob = exp2prob(prediktiontemp');
    prob = prob(:,1);
    odds2 = 1./odds1;
    probodds = odds2./(sum(odds2));
    sannolikhet=[sannolikhet; prob];

    probvec = sort(prob(:,1),'descend')';
    jmntvec = [];
    for k = 1:length(probvec)
        jmntvec = [jmntvec sum(probvec(1:k))];
    end

    jmntvec=[jmntvec zeros(1,15-length(jmntvec))];
    jmntmatris = [jmntmatris; jmntvec];
    antalhorses = [antalhorses length(rank)];
    jmntvec=[jmntvec zeros(1,15-length(jmntvec))];
    jmntmatris = [jmntmatris; jmntvec];
    antalhorses = [antalhorses length(rank)];
    probvec15 = [probvec zeros(1,15-length(probvec))];
    probmatris = [probmatris; probvec15];
    probunsortedtemp = [prob(:,1)' zeros(1,15-length(prob
      (:,1)))];
    probunsorted = [probunsorted; probunsortedtemp];
```

```
    proboddstemp = [probodds' zeros(1,15-length(probodds(:,1)))];
    proboddsmatris = [proboddsmatris; proboddstemp ];
    oddstemp = [odds1' zeros(1,15-length(odds1))];
    oddsmatris = [oddsmatris; oddstemp];

    idmatris(1:length(predictionhorse.id(plats(j):plats(j+1)-1)),j) =   predictionhorse.id(plats(j):plats(j+1)-1);
    rankpred=nontiedrank(prediktion(plats(j):plats(j+1)-1));
    rankodds=nontiedrank(odds(plats(j):plats(j+1)-1));
    jmnt2 = [jmnt2; prediktiontemp(find(rankpred==2)) - prediktiontemp(find(rankpred==1))];
    jmnt3 = [jmnt3; prediktiontemp(find(rankpred==3)) - prediktiontemp(find(rankpred==1))];
    jmnt4 = [jmnt4; prediktiontemp(find(rankpred==4)) - prediktiontemp(find(rankpred==1))];
    jmnt5 = [jmnt5; prediktiontemp(find(rankpred==5)) - prediktiontemp(find(rankpred==1))];
    jmnt6 = [jmnt6; prediktiontemp(find(rankpred==6)) - prediktiontemp(find(rankpred==1))];
    jmnt7 = [jmnt7; prediktiontemp(find(rankpred==7)) - prediktiontemp(find(rankpred==1))];
    predrankvinnare = [predrankvinnare; rank(1)];
    oddsrankvinnare = [oddsrankvinnare; tiedrankodds(1)];
end
optimeringspelsystem
```

```
%% Optimizing V75
load vec_400-500 % Loading different combinations

vec = [];
antalhorses=[10 12 13 14 15 12 12];
for a = 1:min(antalhorses(1), 9)
    for b = 1:min(antalhorses(2), 9)
        for c = 1:min(antalhorses(3), 9)
            for d = 1:min(antalhorses(4), 9)
                for e = 1:min(antalhorses(5), 9)
                    for f = 1:min(antalhorses(6), 9)
                        for g = 1:min(antalhorses(7), 9)
                            if a*b*c*d*e*f*g >=400*2 && a*b*c*d*e*f*g <= 500*2
                                vec = [vec; a b c d e f g];
                            end
                        end
                    end
                end
            end
```

```
        end
    end
end

vinstvec = [];vinstvec04 = [];vinstvec05 = [];vinstvec06 = [];vinstvec07 = [];

for j=1:length(vec)
    vinstvec= [vinstvec; jmntmatris(1,vec(j,1))*jmntmatris(2,vec(j,2))*...
    jmntmatris(3,vec(j,3))*jmntmatris(4,vec(j,4))*jmntmatris(5,vec(j,5))*...
    jmntmatris(6,vec(j,6))*jmntmatris(7,vec(j,7))];
    A = [jmntmatris(1,vec(j,1)) jmntmatris(2,vec(j,2)) jmntmatris(3,vec(j,3))   jmntmatris(4,vec(j,4))...
    jmntmatris(5,vec(j,5)) jmntmatris(6,vec(j,6)) jmntmatris(7,vec(j,7))];
    index = find(A>=0.4);

    numberOfElements = length(index);
    if numberOfElements > 6
        vinstvec04 = [vinstvec04; jmntmatris(1,vec(j,1))*jmntmatris(2,vec(j,2))*...
        jmntmatris(3,vec(j,3))*jmntmatris(4,vec(j,4))*jmntmatris(5,vec(j,5))*...
        jmntmatris(6,vec(j,6))*jmntmatris(7,vec(j,7))];
    else
    vinstvec04 = [vinstvec04; 0];
    end

    index = find(A>=0.5);
    numberOfElements = length(index);
    if numberOfElements > 6
        vinstvec05 = [vinstvec05; jmntmatris(1,vec(j,1))*jmntmatris(2,vec(j,2))*...
        jmntmatris(3,vec(j,3))*jmntmatris(4,vec(j,4))*jmntmatris(5,vec(j,5))*...
        jmntmatris(6,vec(j,6))*jmntmatris(7,vec(j,7))];
    else
    vinstvec05 = [vinstvec05; 0];
    end

    index = find(A>=0.6);
    numberOfElements = length(index);
    if numberOfElements > 6
        vinstvec06 = [vinstvec06; jmntmatris(1,vec(j,1))*jmntmatris(2,vec(j,2))*...
        jmntmatris(3,vec(j,3))*jmntmatris(4,vec(j,4))*jmntmatris(5,vec(j,5))*...
```

```
            jmntmatris(6,vec(j,6))*jmntmatris(7,vec(j,7))];
    else
        vinstvec06 = [vinstvec06; 0];
    end

    index = find(A>=0.7);
    numberOfElements = length(index);
    if numberOfElements > 6
        vinstvec07 = [vinstvec07; jmntmatris(1,vec(j,1))*
          jmntmatris(2,vec(j,2))*...
        jmntmatris(3,vec(j,3))*jmntmatris(4,vec(j,4))*
          jmntmatris(5,vec(j,5))*...
        jmntmatris(6,vec(j,6))*jmntmatris(7,vec(j,7))];
    else
        vinstvec07 = [vinstvec07; 0];
    end
end

% Win often
slh= max(vinstvec);slh04 = max(vinstvec04);slh05 = max(
  vinstvec05);slh06 = max(vinstvec06);
slh07 = max(vinstvec07);a=find(vinstvec==slh);spel=vec(a,:)
  ;
if slh04 > 0
    a=find(vinstvec04==slh04);
    spel04=vec(a,:);
else
    spel04 =[];
end

if slh05 > 0
    a=find(vinstvec05==slh05);
    spel05=vec(a,:);
else
    spel05 =[];
end

if slh06 > 0
    a=find(vinstvec06==slh06);
    spel06=vec(a,:);
else
    spel06 =[];
end

if slh07 > 0
    a=find(vinstvec07==slh07);
    spel07=vec(a,:);
else
    spel07 =[];
```

```
end
```

```
%% Model 3: Decide wich covariates to include based on ROC

load('minisuperhorse10')%Load data set
load('predictionhorse10')

Y = zeros(length(minisuperhorse.resultat),1);

for j = 1:length(Y)
    if minisuperhorse.resultat(j)==1
        Y(j)=1;
    end
end

AUCvecPRED =[];maxAUCvec = [];

covariates=[{'odds'} {'pengar'} {'alder' 'brakusk'} {'
  mycketbrakusk'} {'ganskabrakusk'} {'mktbrahorse'}...
{'ganskabrahorse'} {'brahorse'} {'sto'} {'hingst'} {'
  vinstprocent'} {'platsprocent'}...
{'startp'} {'vinstform'} {'segerform'} {'mintid'} {'
  StartnummerRank'}...
{'distansvinsttrivsel'} {'distanstopp3trivsel'} {'
  sommarvinsttrivsel'} {'sommartopp3trivsel'}...
{'autovinsttrivsel'} {'autotopp3trivsel'} {'distansdummy'}
  {'autostart'}]';

covatiatematrixReg = [minisuperhorse.odds minisuperhorse.
  pengar minisuperhorse.alder...
minisuperhorse.brakusk minisuperhorse.mycketbrakusk
  minisuperhorse.ganskabrakusk...
minisuperhorse.mktbrahorse minisuperhorse.ganskabrahorse
  minisuperhorse.brahorse...
minisuperhorse.sto minisuperhorse.hingst minisuperhorse.
  vinstprocent...
minisuperhorse.platsprocent minisuperhorse.startp
  minisuperhorse.vinstform...
minisuperhorse.segerform' minisuperhorse.mintid
  minisuperhorse.StartnummerRank...
minisuperhorse.distansvinsttrivsel minisuperhorse.
  distanstopp3trivsel...
minisuperhorse.sommarvinsttrivsel minisuperhorse.
  sommartopp3trivsel...
minisuperhorse.autovinsttrivsel minisuperhorse.
  autotopp3trivsel...
minisuperhorse.distansdummy minisuperhorse.autostart];

```

```
RMSpred = [];RMSvecdata = [];RMSvecpred = [];Xreg = [];
  squared = [];
sqroot = [];squaredcov = [];sqrootcov = [];multcov = [];
  alonecov = [];
antcov = length(covatiatematrixReg(1,:)); ant = 1;vek = [];

for k = 1:antcov
    alonecov = [alonecov; covariates(k) {'alone'}];
    squared = [squared covatiatematrixReg(:,k).^2];
    squaredcov = [squaredcov; covariates(k) {'squared'}];
    sqroot = [sqroot covatiatematrixReg(:,k).^1/2];
    sqrootcov = [sqrootcov; covariates(k) {'root'}];

    for n = k+1:antcov
        if sum(covatiatematrixReg(:,k).*covatiatematrixReg
          (:,n)) ~=0
            Xreg=[Xreg covatiatematrixReg(:,k).*
              covatiatematrixReg(:,n)];
            multcov = [multcov; covariates(k) covariates(n)
              ];
        end
    end
end

finalcovariates = [alonecov; squaredcov ;sqrootcov; multcov
  ;];
covatiatematrixReg = [covatiatematrixReg sqroot squared];

covatiatematrixPred = [predictionhorse.odds predictionhorse
  .pengar predictionhorse.alder...
predictionhorse.brakusk predictionhorse.mycketbrakusk
  predictionhorse.ganskabrakusk...
predictionhorse.mktbrahorse predictionhorse.ganskabrahorse
  predictionhorse.brahorse...
predictionhorse.sto predictionhorse.hingst predictionhorse.
  vinstprocent...
predictionhorse.platsprocent predictionhorse.startp
  predictionhorse.vinstform...
predictionhorse.segerform predictionhorse.mintid
  predictionhorse.StartnummerRank...
predictionhorse.distansvinsttrivsel predictionhorse.
  distanstopp3trivsel...
predictionhorse.sommarvinsttrivsel predictionhorse.
  sommartopp3trivsel...
predictionhorse.autovinsttrivsel predictionhorse.
  autotopp3trivsel...
predictionhorse.distansdummy predictionhorse.autostart];

Xpred=[];squared = [];sqroot1 = [];
```

```
for t = 1:antcov
    squared = [squared covatiatematrixPred(:,t).^2];
    sqroot1 = [sqroot1 covatiatematrixPred(:,t).^1/2];

    for n = t+1:antcov
        if sum(covatiatematrixReg(:,t).*covatiatematrixReg(:,n)) ~=0
            Xpred=[Xpred covatiatematrixPred(:,t).*covatiatematrixPred(:,n)];
        end
    end
end

covatiatematrixPred = [covatiatematrixPred sqroot1 squared];

AUCvec=[];used =[];
[rader kolonner] = size(covatiatematrixReg);
XregFINAL = covatiatematrixReg(:,used);
XpredFINAL = [];RMSdata=[];
okindex = 1:kolonner;

for k = 1:300
    AUCvec=[];
    for m =okindex
        Xreg = XregFINAL;
        antkol=m;
        Xreg=[Xreg covatiatematrixReg(:,m)];
        b = glmfit(Xreg,Y,'binomial','link','logit');
        Xpred=[covatiatematrixPred(:,used) covatiatematrixPred(:,m)];
        prediktion = exp([ones(length(predictionhorse.odds),1)      Xpred]*b)./(1+exp([ones(length(predictionhorse.odds),1) Xpred]*b));

        a1 = predictionhorse.resultat;
        b1 = prediktion;

        Y1= zeros(length(a1),1);
        for o=1:length(a1)
            if a1(o)==1
                Y1(o)=1;
            end
        end

        p = glmval(b,Xreg,'logit');
        if sum(find(b==0))>=1
            AUC=0;
```

```
        else
            [x,y,T,AUC,OPTROCPT,SUBY,SUBYNAMES]=perfcurve(Y
              ,p,1); % Calc AUC
        end

        AUCvec(m)=AUC;
    end

    a=find(AUCvec ==max(AUCvec));
    used =[used;a(1)];
    XregFINAL = [XregFINAL covatiatematrixReg(:,a(1))];
    b = glmfit(XregFINAL,Y,'binomial','link','logit');
    Xpred=covatiatematrixPred(:,used);
    prediktion = exp([ones(length(predictionhorse.odds),1)
       Xpred]*b)./(1+exp([ones(length(predictionhorse.odds)
      ,1) Xpred]*b));
    [x,y,T,AUC,OPTROCPT,SUBY,SUBYNAMES]=perfcurve(Y1,
      prediktion,1);
    AUCvecPRED=[AUCvecPRED; AUC];
    SSEpred=sum((Y1-prediktion).^2);
    n=length(Y1);
    RMSvecpred=[RMSvecpred; sqrt(SSEpred/n)];
    p = glmval(b,XregFINAL,'logit');
    SSEdata=sum((Y-p).^2);
    n=length(Y);
    RMSdata = [RMSdata; sqrt(SSEdata/n)];
    okindex(find(okindex==a(1)))=[];
    maxAUCvec = [maxAUCvec; max(AUCvec)]
    [RMSdata(end) RMSvecpred(end) used(end)]
end
toc
```

```
%% Model 3
close all, clc, clear all

% Data
load predictionhorse10
load minisuperhorse10
load used

% Logistic result
minisuperhorse.logisticresultat=zeros(length(minisuperhorse
  .resultat),1);
plats=find(minisuperhorse.resultat==1);
minisuperhorse.logisticresultat(plats)=1;

% Dependent variable
Y = minisuperhorse.logisticresultat;

```

```
% Covariates
covatiatematrixReg = [minisuperhorse.odds minisuperhorse.pengar minisuperhorse.alder...
minisuperhorse.brakusk minisuperhorse.mycketbrakusk minisuperhorse.ganskabrakusk...
minisuperhorse.mktbrahorse minisuperhorse.ganskabrahorse minisuperhorse.brahorse...
minisuperhorse.sto minisuperhorse.hingst minisuperhorse.vinstprocent...
minisuperhorse.platsprocent minisuperhorse.startp minisuperhorse.vinstform...
minisuperhorse.segerform' minisuperhorse.mintid minisuperhorse.StartnummerRank...
minisuperhorse.distansvinsttrivsel minisuperhorse.distanstopp3trivsel...
minisuperhorse.sommarvinsttrivsel minisuperhorse.sommartopp3trivsel...
minisuperhorse.autovinsttrivsel minisuperhorse.autotopp3trivsel...
minisuperhorse.distansdummy minisuperhorse.autostart];

covariates=[{'odds'} {'pengar'} {'alder'} {'brakusk'} {'mycketbrakusk'} {'ganskabrakusk'} {'mktbrahorse'}...
{'ganskabrahorse'} {'brahorse'} {'sto'} {'hingst'} {'vinstprocent'} {'platsprocent'}...
{'startp'} {'vinstform'} {'segerform'} {'mintid'} {'StartnummerRank'}...
{'distansvinsttrivsel'} {'distanstopp3trivsel'} {'sommarvinsttrivsel'} {'sommartopp3trivsel'}...
{'autovinsttrivsel'} {'autotopp3trivsel'} {'distansdummy'} {'autostart'}]';

% X Regression
Xreg = [];squared = [];sqroot = [];
squaredcov = [];sqrootcov = [];multcov = [];alonecov = [];
antcov = length(covatiatematrixReg(1,:));ant = 1;vek = [];

for k = 1:antcov
    alonecov = [alonecov; covariates(k) {'alone'}];
    squared = [squared covatiatematrixReg(:,k).^2];
    squaredcov = [squaredcov; covariates(k) {'squared'}];
    sqroot = [sqroot covatiatematrixReg(:,k).^1/2];
    sqrootcov = [sqrootcov; covariates(k) {'root'}];
    for n = k+1:antcov
        if sum(covatiatematrixReg(:,k).*covatiatematrixReg(:,n)) ~=0
            Xreg=[Xreg covatiatematrixReg(:,k).*covatiatematrixReg(:,n)];
```

```
                multcov = [multcov; covariates(k) covariates(n)];
            end
        end
end

finalcovariates = [alonecov; squaredcov ;sqrootcov; multcov;];
covatiatematrixReg = [covatiatematrixReg sqroot squared Xreg];
X=covatiatematrixReg(:,used);

% Regression
b = glmfit(X,Y,'binomial','link','logit');

% X Prediction
covatiatematrixPred = [predictionhorse.odds predictionhorse.pengar predictionhorse.alder...
predictionhorse.brakusk predictionhorse.mycketbrakusk predictionhorse.ganskabrakusk...
predictionhorse.mktbrahorse predictionhorse.ganskabrahorse predictionhorse.brahorse...
predictionhorse.sto predictionhorse.hingst predictionhorse.vinstprocent...
predictionhorse.platsprocent predictionhorse.startp predictionhorse.vinstform...
predictionhorse.segerform predictionhorse.mintid predictionhorse.StartnummerRank...
predictionhorse.distansvinsttrivsel predictionhorse.distanstopp3trivsel...
predictionhorse.sommarvinsttrivsel predictionhorse.sommartopp3trivsel...
predictionhorse.autovinsttrivsel predictionhorse.autotopp3trivsel...
predictionhorse.distansdummy predictionhorse.autostart];

Xpred=[];squared = [];sqroot1 = [];
for t = 1:antcov
    squared = [squared covatiatematrixPred(:,t).^2];
    sqroot1 = [sqroot1 covatiatematrixPred(:,t).^1/2];
    for n = t+1:antcov
        if sum(covatiatematrixReg(:,t).*covatiatematrixReg(:,n)) ~=0
            Xpred=[Xpred covatiatematrixPred(:,t).*covatiatematrixPred(:,n)];
        end
    end
end

```

```
covatiatematrixPred = [covatiatematrixPred sqroot1 squared
  Xpred];
X=[ones(length(predictionhorse.odds),1) covatiatematrixPred
  (:,used)];

% Prediction
prediktion = exp(X*b)./(1+exp(X*b));

% Analys
kassa=1000;
dagskassa = analysLogistisk(predictionhorse,prediktion,
  kassa);

% Cash plot
m=1000;
k=(dagskassa(end)-m)/length(dagskassa);
kx=1:length(dagskassa);
y=k*kx+m;
plot(kx,dagskassa,kx,dagskassa,'bo',kx,y,'r')
title('Betting strategy 3')
xlabel('Number of races ')
ylabel('SEK')
```

```
function dagskassa = analysLogistisk(predictionhorse,
  prediktion, kassa)
dagskassa=[];

% Real result
resultat = predictionhorse.omvresultat';
odds=predictionhorse.odds;

% Find races
unkiloppid=unique(predictionhorse.loppid);

% Loop for each race
for j=1:length(unkiloppid)
    plats=find(predictionhorse.loppid==unkiloppid(j));
    rank=tiedrank(-prediktion(plats));
    prediktiontemp = prediktion(plats);
    tiedrankodds=tiedrank(odds(plats));
    odds1=odds(plats)/10;
    resultat1=resultat(plats);

    % Predicted prob
    prob = prediktiontemp/(sum(prediktiontemp));

    % Prob odds
    odds2 = 1./odds1;
    probodds = odds2./(sum(odds2));
```

```

    % Disp result
    m=[resultat1' tiedrankodds odds1 rank prediktiontemp prob(:,1) probodds prob(:,1)-odds2];
    [B I]=sort(m(:,1));
    m2=m(I,:);
    disp(m2)

    % Cash
    asd=find((prob-odds2)>0);
    betprob=max(prob(asd))
    if isempty(betprob)==0
    k=find(betprob==prob);
    oddsbet=odds1(k)

    if betprob > 0.2
        summa=200;
    else
        summa=100;
    end

    if betprob>0.1
        placering=resultat1(k)
        if placering==1
            kassa=kassa+summa*(odds1(k)-1);
        else
            kassa=kassa-summa;
        end
    end
end
dagskassa=[dagskassa; kassa];
end
```