

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

1	Introduction	1
1.1	Project	1
1.2	Swedish Harness Racing	1
1.2.1	The Betting System for V75	2
1.2.2	The Betting System for Betting on a Single Horse . . .	2
1.3	Nomenclature	2
2	Theory	5
2.1	The Linear Regression Model	5
2.1.1	Estimation	5
2.1.2	Standard Error of Beta	6
2.1.3	Prediction	6
2.2	The Logistic Regression Model	7
3	Data	8
3.1	Collecting and Sorting the Data	8
3.1.1	Data Used when Running the Regression	8
3.1.2	Data Used when Running the Prediction	9
3.1.3	Problems with the Data	9
4	Modeling	10
4.1	Modeling	10
4.1.1	Data	10
4.1.2	Evaluation of the Models	11
4.2	Covariates	11
4.2.1	Taken From the Data	12

4.2.2	Calculated From the Data	12
4.2.3	Made-up and Calculated From the Data	14
4.3	The Models	15
4.3.1	The First Model	15
4.3.2	The Second Model	17
4.3.3	The Third Model	19
5	Using the Model for Betting	21
5.1	The Win Often Betting Strategy	21
5.2	Betting in a Single Race	22
6	Results	24
6.1	Models	24
6.2	Predictions	25
6.3	Betting	26
7	Discussion	30
7.1	Thoughts	30
7.2	For the Future	30
A	Betting Strategies	33
A.1	The Win Big Betting Strategy	33
A.1.1	Reduced Systems	33
A.1.2	Reduced Systems 2.0	33
B	MATLAB Scripts - Data	35
C	MATLAB Scripts - Models	74

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 $3 \cdot 2 \cdot 1 \cdot \dots \cdot 1 = 6$ number of rows and thus costs three SEK is shown in figure 1.2.1 below.

The screenshot shows a betting interface for V75. At the top, it says 'JÄGERSRO TR LÖRDAG 23 MARS 2013' and 'START 14:30'. Below this, there's a checkbox for 'V7, endast vinst vid 7 rätt'. The main part of the interface consists of seven rows of horse selection, labeled 'AVD 1' through 'AVD 7'. Each row has buttons for horses 1 through 15, and an 'ALLA' button. The selection for each row is as follows: AVD 1: 1, 2, 3; AVD 2: 1, 2, 3; AVD 3: 1, 2, 3; AVD 4: 1, 2, 3; AVD 5: 1, 2, 3; AVD 6: 1, 2, 3; AVD 7: 1, 2, 3. At the bottom, there's a 'NY KUPONG' logo, a 'RADER' section with 'Harry Boy' and 'Reserver' checkboxes, and a 'SYSTEM' section with a '6 x 1 x 0,50' configuration. The total cost is 3,00 SEK, and there are 'RENSA' and 'SPELA' buttons.

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

$$y_i = \sum_{j=0}^k x_{i,j} \beta_j + e_i, \quad i = 1, \dots, n \quad (2.1)$$

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

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

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

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}. \quad (2.3)$$

2.1.1 Estimation

The estimated values of $\boldsymbol{\beta}$ are presented as $\hat{\boldsymbol{\beta}}$. 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

$$E \left[\hat{\boldsymbol{\beta}} \mid \mathbf{X} \right] = \boldsymbol{\beta}, \quad (2.4)$$

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

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

Where an unbiased estimator of the regression variance

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

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

$$E [s^2 | \mathbf{X}] = \sigma^2 \quad (2.7)$$

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

$$Cov(\hat{\beta} | \mathbf{X}) = (\mathbf{X}^t \mathbf{X})^{-1} \sigma^2 \quad (2.8)$$

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

$$\hat{Cov}(\hat{\beta} | \mathbf{X}) = (\mathbf{X}^t \mathbf{X})^{-1} s^2. \quad (2.9)$$

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

2.1.3 Prediction

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

$$y_p = \mathbf{x}_0 \hat{\beta}, \quad (2.10)$$

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. These 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

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

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 `transport.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 where all horses are available. The biggest problem was that during the project `transport.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

$$AIC = 2k - 2\ln(L) \quad (4.1)$$

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 worsen 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, $P(\text{horse predicted as winner does not win})$ and the y-axis is defined as $P(\text{horse predicted as winner wins})$. 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

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

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

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

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

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

From Table 4.1 it can be seen that when a covariate, in this case *Age*, 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.

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

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	AIC _{REG}	RMSE _{REG}
<i>Odds</i>	149582.3	0.800
<i>Odds + Win shape</i>	149317.8	0.799
<i>Odds + Win shape + Age</i>	149319.8	0.799
<i>Odds + Win shape + Win percentage</i>	149150.2	0.797
\vdots	\vdots	\vdots
<i>Odds + Win shape + ... + Start number rank</i>	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

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

So the placement is predicted as

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

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
\vdots	\vdots	\vdots
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 overparameterization. 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.

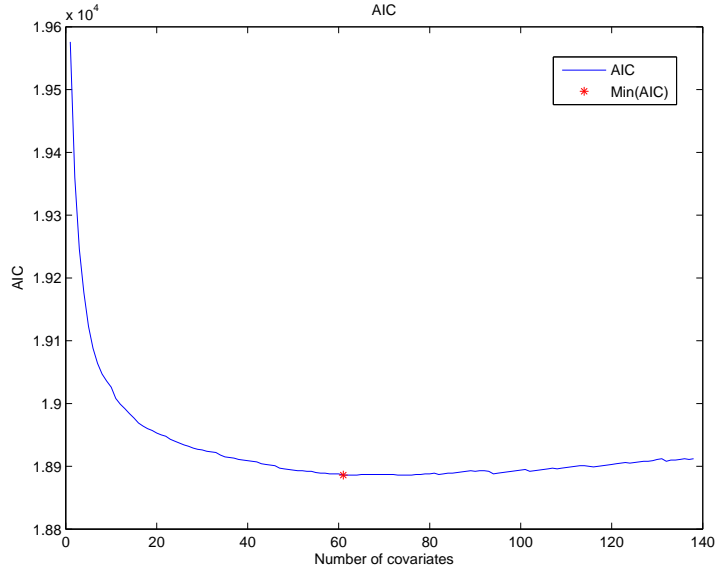


Figure 4.3.1: Plot of AIC.

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

$$\ln(\text{placement}_i) = \beta_0 + \text{Odds}_i \cdot \beta_1 + \dots + \text{Time}_i \cdot \beta_k + e_i, \quad i = 1, \dots, n. \quad (4.5)$$

The placement can now be predicted as

$$\text{placement}_i = \exp\left(\hat{\beta}_0 + \dots + \text{Time}_i \cdot \hat{\beta}_k\right), \quad i = 1, \dots, n. \quad (4.6)$$

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.	AUC _{REG}	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.

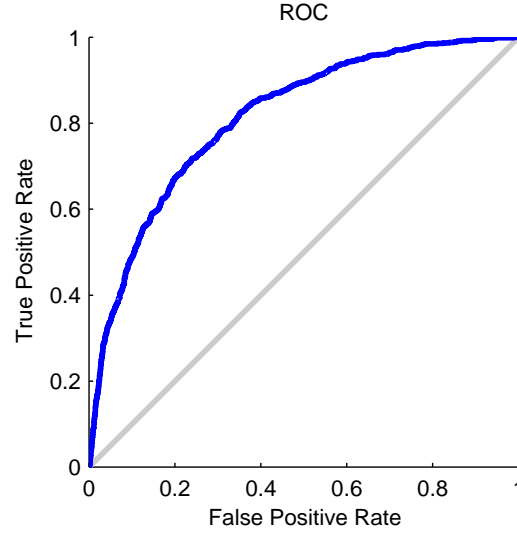


Figure 4.3.2: Plot of ROC.

Model three

$$y_i = \frac{\exp(\beta_0 + \dots + Time_i \cdot \beta_k)}{1 + \exp(\beta_0 + \dots + Time_i \cdot \beta_k)}, \quad i = 1, \dots, n \quad (4.7)$$

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

$$p_i = \frac{\exp(\hat{\beta}_0 + \dots + Time_i \cdot \hat{\beta}_k)}{1 + \exp(\hat{\beta}_0 + \dots + Time_i \cdot \hat{\beta}_k)}, \quad i = 1, \dots, n. \quad (4.8)$$

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

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

under the condition that

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

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

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

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

$$p_{\text{odds}} = \frac{1}{\text{odds}}. \quad (5.4)$$

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

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

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.

$$p_j - \tilde{p}_{odds_j} \geq \lambda. \quad (5.6)$$

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

$$E[\text{money won}] = p_j \cdot (\text{odds}_j - 1) \cdot x \quad (5.7)$$

$$E[\text{money lost}] = (1 - p_j) \cdot x. \quad (5.8)$$

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

$$E[\text{money won}] > E[\text{money lost}] \iff p_j > \frac{1}{\text{odds}_j} = p_{odds_j}. \quad (5.9)$$

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, $b \rightarrow \infty$, you will make a profit. But if equation 5.3 is met combined with a criterion on the probability like

$$p_j \geq \mu \quad (5.10)$$

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

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

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

Model	RMSE_{REG}	RMSE_{PRED}
<i>Model 2</i>	0.2840	0.2627
<i>Model 3</i>	0.2848	0.2644
<i>Odds</i>	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	<i>Jackpot</i>	−960
3	3	500	0	−1460
4	6	480	48	−1892
5	7	486	1966	−412
6	5	480	<i>Jackpot</i>	−912
7	5	480	<i>Jackpot</i>	−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	<i>Jackpot</i>	−972
3	3	500	0	−1472
4	5	500	<i>Jackpot</i>	−1972
5	5	486	<i>Jackpot</i>	−2458
6	6	486	21	−2923
7	5	486	<i>Jackpot</i>	−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

$$p_j > p_{odds_j}, \quad (6.2)$$

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%.

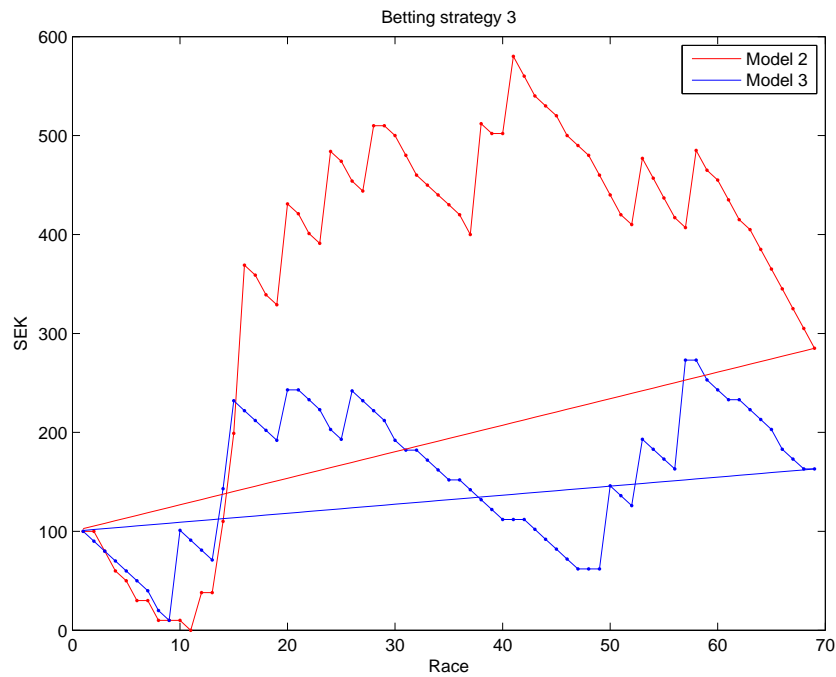


Figure 6.3.1: Result with betting strategy 3. Both models will result in a increase of the bankroll. Also notice that the large increase of the bankroll at the left of the plot is not just one but several wins.

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 `transport.se` removes foreign registered horses a few days after a race, fu-

ture 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

```
1 %% Script to find the id of horses a V75 race day
2
3 clc, close all, clear all
4 tic
5
6 URLvec = []; antalhorses = []; horsevec = [];
7
8 tevdagID='529896'; % A given race day
9 url= ['https://www.travsport.se/sresultat?kommando=
10      tevlingsdagVisa&tevdagId=' tevdagID];
11 urlwrite(url, 'test.txt');
12
13 filnamn = ['test.txt'];
14 fid = fopen(filnamn,'rt');
15 C=textread(filnamn, '%s', 'delimiter', '\n');
16
17 j = 1; n = 1;
18 hitta = 'V75-1'; % Find every V75 race
19
20 while j < length(C)
21     rad = C{j};
22
23     if strfind(rad, hitta)
24         j = j + 1;
25         rad = C{j};
26         start = length('<a href="') + 1;
27         stop = strfind(rad, '" class="large" >') - 1;
28         n = n + 1;
29         hitta = ['V75-' num2str(n)];
30         URLvec = [URLvec; {num2str(rad(start:stop))}];
31     elseif strfind(rad, 'Strukna')
32         j=length(C);
```



```

33     end
34     j = j+1;
35 end
36 fclose(fid);
37
38 % Find the horses
39 hitta = '<a href="/hast/visa/';
40
41 for i = 1:length(URLvec)
42     url = ['https://www.travsport.se' URLvec{i}];
43     pause(1)
44     urlwrite(url, 'test.txt');
45     fid = fopen(filnamn,'rt');
46     C=textread(filnamn, '%s', 'delimiter', '\n');
47     n = 0;
48     j = 1;
49     while j < length(C)
50         rad = C{j};
51         if strfind(rad, hitta)
52             start = length(hitta) + 1;
53             stop = strfind(rad, '/resultat') - 1;
54             horsevec = [horsevec; {(rad(start:stop))}];
55             n = n + 1;
56         elseif strfind(rad, 'Strukna')
57             j=length(C);
58         end
59         j = j + 1;
60     end
61     fclose(fid);
62
63     antalhorses = [antalhorses n];
64 end
65 save('antalhorses','antalhorses');
66 save('horsevec','horsevec'); %Saves a vector of horses
67 participating that day
68 delete('C:\Users\mhella\Documents\MATLAB\KEX\NY\test.txt');
69 InmatningTillModeleringsprogrammet
70 loppinfo2_martinTillmodelleringINKLutlandskalopp
71 toc

```

```

1 %% Script to download HTML from travsport.se
2
3 load('horsevec') %A given vector of horses
4
5 for n = 1:length(horsevec)
6     filnamn = [num2str(horsevec{n}) '.txt'];
7     url= ['https://www.travsport.se/hast/visa/' num2str(
        horsevec{n}) '/resultat'];

```

```

8      try
9          bla = urlwrite(url, filnamn);
10     catch err
11         disp(num2str(n)) %Catches error if theres no horse
12         with that id
13     end
14 end

1 %% Textfile to struct
2
3 clc, close all, clear all
4 tic
5 hej = 0;
6
7 % STRUCT
8 horse.namn = [];horse.id = [];horse.color = [];horse.gender
   = [];
9 horse.birthdate = [];horse.ras = [];horse.loppid = [];horse
   .datum = [];
10 horse.startnummer = [];horse.loppnummer = [];horse.distan
   s = [];
11 horse.tid = [];horse.resultat=[];horse.ntungbana = [];horse
   .odds = [];
12 horse.kusk = [];horse.trainer = [];horse.vinterbana = [];
13 horse.tungbana = [];horse.autostart = [];horse.galopp = [];
14 horse.skorfram = [];horse.skorbak = [];horse.vinst = [];
   horse.bana=[];
15 horse_id = [];
16
17 load('horsevec'); %Loading vector of downloaded horses
18
19 ok = 1;
20 i = 1;
21
22 for h = 1:length(horsevec)
23     n = horsevec{h};
24     check = 1;
25     try
26         filnamn = [num2str(n) '.txt'];
27         fid = fopen(filnamn,'rt');
28         C=textread(filnamn, '%s', 'delimiter', '\n');
29     catch err
30         check = 0;
31     end
32     if check == 1
33         for j = 1:length(C)
34             rad = C{j};
35             okej4=1;
36

```

```

37      % Name
38      if j == 460
39          start = length('<span class="nottranslate">')
40              +1;
41          stop = strfind(rad, '</span> <br/>') - 1;
42          horse(i).namn = rad(start:stop);
43
44      % Color
45      elseif j == 465
46          start = 5;
47          stop = strfind(rad, '</td>') - 1;
48          horse(i).color = rad(start:stop);
49
50      % Gender
51      elseif j == 466
52          start = 14;
53          stop = strfind(rad, '</td>') - 1;
54          horse(i).gender = rad(start:stop);
55
56      % Birthdate
57      elseif j == 467
58          horse(i).birthdate = str2double([rad(14:17)
59              rad(19:20) rad(22:23)]);
60
61      % Race
62      elseif j == 468
63          start = 5;
64          stop = strfind(rad, '</td>') - 1;
65          horse(i).ras = rad(start:stop);
66      end
67
68      if isempty(strfind(rad, '130504')) == 0
69          ok = 1;
70
71      % Racetrack
72      radbana=C{j-1};
73      start = 5;
74      stop = strfind(radbana, '</td>') - 1;
75      horse(i).bana = [horse(i).bana; {radbana(
76          start:stop)}];
77      radstart=C{j+2};
78
79      % Starting number
80      start = length('<td colspan="2" class="
81          right_align"><span>') + 1;
82      slash = strfind(radstart(start:end), '/');
83      stop = start - 1 + slash(1) - 1;
84      horse(i).startnummer = [horse(i).
85          startnummer; str2double(radstart(start:

```

```

81         stop))];
82
83         % Date
84         horse(i).datum= [horse(i).datum; 130504];
85
86         % Distance
87         start = stop + 2;
88         stop = start - 1 + strfind(radstart(start:
89             end), '</span></td>') - 1;
90         horse(i).distans= [horse(i).distans;
91             str2double(radstart(start:stop))];
92
93         % Driver
94         radkusk = C{j+14};
95         kuskstart = strfind(radkusk,'>') + 1;
96         kuskstop = strfind(radkusk,'</') - 1;
97         horse(i).kusk = [horse(i).kusk; {radkusk(
98             kuskstart:kuskstop)}];
99
100         horse(i).loppid = [horse(i).loppid; 0];
101         horse(i).resultat = [horse(i).resultat; 0];
102         horse(i).loppnummer = [horse(i).loppnummer;
103             0];
104         horse(i).tid = [horse(i).tid; 0];
105         horse(i).ntungbana = [horse(i).ntungbana;
106             0];
107         horse(i).trainer = [horse(i).trainer; {0}];
108         horse(i).skorbak = [horse(i).skorbak; 0];
109         horse(i).skorfram = [horse(i).skorfram; 0];
110         horse(i).vinst = [horse(i).vinst; 0];
111         horse(i).galopp = [horse(i).galopp; 0];
112         horse(i).tungbana = [horse(i).tungbana; 0];
113         horse(i).vinterbana = [horse(i).vinterbana;
114             0];
115         horse(i).autostart = [horse(i).autostart;
116             0];
117     end
118
119     if isempty(strfind(rad, 'loppId')) == 0
120         ok = 1;
121
122         % Race id
123         start = strfind(rad, 'loppId') + 1 + length
124             ('loppId');
125         stop = start - 1 + strfind(rad(start:end),
126             '><span>') - 1;
127         horse(i).loppid = [horse(i).loppid;
128             str2double(rad(start:stop))];

```

```

119         % Racetrack
120         radbana=C{j-1};
121         start = 5;
122         stop = strfind(radbana, '</td>') - 1;
123         horse(i).bana = [horse(i).bana; {radbana(
124             start:stop)}];
125
126         % Date
127         start = stop + length('><span>') + 1;
128         stop = start - 1 + strfind(rad(start:end),
129             '-') - 1;
130         horse(i).datum = [horse(i).datum;
131             str2double(rad(start:stop))];
132
133         % LOPPNUMMER
134         start = stop + 2;
135         stop = start - 1 + strfind(rad(start:end),
136             '</span>') - 1;
137         horse(i).loppnummer = [horse(i).loppnummer;
138             str2double(rad(start:stop))];
139
140         elseif isempty(strfind(rad, '<td class="nowrap
141             "><span><em><span>')) == 0
142             ok = 1;
143
144         % Race id
145         horse(i).loppid = [horse(i).loppid; 0];
146
147         % Date
148         start = strfind(rad, '<td class="nowrap"><
149             span><em><span>') + length('<td class="
150             nowrap"><span><em><span>');
151         stop = strfind(rad, '-') - 1;
152         horse(i).datum = [horse(i).datum;
153             str2double(rad(start:stop))];
154
155         % Race number
156         start = stop + 2;
157         stop = start - 1 + strfind(rad(start:end),
158             '</span></em></span></td>') - 1;
159         horse(i).loppnummer = [horse(i).loppnummer;
160             str2double(rad(start:stop))];
161     end
162
163     if isempty(strfind(rad, '<td colspan="2" class
164         ="right_align"><span>')) == 0 && length(horse(
165         i).datum) > length(horse(i).
166         startnummer)

```

```

154      % Result
155      radresultat = C{j+1};
156      resultatindex = strfind(radresultat, '>') +
157      1;
158      resultat = radresultat(resultatindex(1));
159
160      if isnan(str2double(resultat)) == 1 &&
161      strcmp(resultat, 'd')==0
162          ok = 0;
163          horse(i).loppid(end) = [];
164          horse(i).datum(end) = [];
165          horse(i).loppnummer(end) = [];
166      elseif isnan(str2double(resultat)) == 1 &&
167      strcmp(resultat, 'd') == 1
168          hej = hej+1;
169          horse(i).resultat = [horse(i).resultat;
170          0];
171      else
172          horse(i).resultat = [horse(i).resultat;
173          str2double(resultat)];
174      end
175
176      if ok == 1
177          %horse(i).resultat = [horse(i).resultat
178          ; resultat];
179
180      % Starting number
181      start = length('<td colspan="2" class="
182      right_align"><span>') + 1;
183      slash = strfind(rad(start:end), '/');
184      stop = start - 1 + slash(1) - 1;
185      horse(i).startnummer = [horse(i).
186      startnummer; str2double(rad(start:stop
187      ))];
188
189      % Distance
190      start = stop + 2;
191      stop = start - 1 + strfind(rad(start:
192      end), '</span></td>') - 1;
193
194      % Track status
195      if strcmp(rad(stop), 'n')
196          stop = stop - 1;
197          horse(i).ntungbana = [horse(i).
198          ntungbana; 1];
199          horse(i).vinterbana = [horse(i).
200          vinterbana; 0];
201          horse(i).tungbana = [horse(i).
202          tungbana; 0];

```

```

190         elseif strcmp(rad(stop), 'v')
191             stop = stop - 1;
192             horse(i).ntungbana = [horse(i).
193                 ntungbana; 0];
194             horse(i).vinterbana = [horse(i).
195                 vinterbana; 1];
196             horse(i).tungbana = [horse(i).
197                 tungbana; 0];
198         elseif strcmp(rad(stop), 't')
199             stop = stop - 1;
200             horse(i).ntungbana = [horse(i).
201                 ntungbana; 0];
202             horse(i).vinterbana = [horse(i).
203                 vinterbana; 0];
204             horse(i).tungbana = [horse(i).
205                 tungbana; 1];
206         else
207             horse(i).ntungbana = [horse(i).
208                 ntungbana; 0];
209             horse(i).vinterbana = [horse(i).
210                 vinterbana; 0];
211             horse(i).tungbana = [horse(i).
212                 tungbana; 0];
213         end
214         horse(i).distans = [horse(i).distans;
215             str2double(rad(start:stop))];
216
217         % Time
218         rattid = C{j+3};
219         tidstop = strfind(rattid, '/');
220
221         % Galop and car start
222         if strcmp(rattid(tidstop-2), 'g') == 1
223             && strcmp(rattid(tidstop-3), 'a') == 1
224                 horse(i).galopp = [horse(i).galopp;
225                     1];
226                 horse(i).autostart = [horse(i).
227                     autostart; 1];
228                 tidstop=tidstop-4;
229             elseif strcmp(rattid(tidstop-2), 'g')
230                 == 1
231                 horse(i).galopp = [horse(i).galopp;
232                     1];
233                 horse(i).autostart = [horse(i).
234                     autostart; 0];
235                 tidstop=tidstop-3;
236             elseif strcmp(rattid(tidstop-2), 'a')
237                 == 1

```

```

221         horse(i).autostart = [horse(i).
222             autostart; 1];
223         horse(i).galopp = [horse(i).galopp;
224             0];
225         tidstop=tidstop-3;
226     else
227         horse(i).autostart = [horse(i).
228             autostart; 0];
229         horse(i).galopp = [horse(i).galopp;
230             0];
231         tidstop=tidstop-2;
232     end
233     tid=[radtid(5:tidstop-2) ',' radtid(
234         tidstop-1:tidstop)];
235     horse(i).tid = [horse(i).tid;
236         str2double(tid)];
237
238     % Odds
239     radodds = C{j+4};
240     oddsstop = strfind(radodds, '</span></
241         td>');
242     odds = radodds(31:oddsstop-1);
243     horse(i).odds = [horse(i).odds;
244         str2double(odds)];
245
246     % Shoes
247     radsko = C{j+8};
248     skostop = strfind(radsko, 'gif"');
249     if isempty(skostop) == 1
250         horse(i).skorfram = [horse(i).
251             skorfram; 0];
252         horse(i).skorbak = [horse(i).
253             skorbak; 0];
254     else
255         if strcmp(radsko(skostop(1)-5), '/')
256             == 1
257             horse(i).skorfram = [horse(i).
258                 skorfram; 1];
259         else
260             horse(i).skorfram = [horse(i).
261                 skorfram; 0];
262         end
263         if strcmp(radsko(skostop(2)-5), '/')
264             == 1
265             horse(i).skorbak = [horse(i).
266                 skorbak; 1];
267         else
268             horse(i).skorbak = [horse(i).
269                 skorbak; 0];

```



```

254         end
255     end
256
257     % Driver
258     radkusk = C{j+12};
259     kuskstart = strfind(radkusk, '>') + 1;
260     kuskstop = strfind(radkusk, '</') - 1;
261     horse(i).kusk = [horse(i).kusk; {
        radkusk(kuskstart:kuskstop)}]];
262
263     % Trainer
264     radtrainer = C{j+16};
265     trainerstart = strfind(radtrainer, '>')
        + 1;
266     trainerstop = strfind(radtrainer, '</')
        - 1;
267     horse(i).trainer = [horse(i).trainer; {
        radtrainer(trainerstart:trainerstop)
        }]];
268
269     % Money
270     radvinst = C{j+20};
271     start = length('<td class="right_align
        nowrap"><span>') + 1;
272     stop = strfind(radvinst, '</span></td>')
        - 1;
273     vinst = radvinst(start:stop);
274     if isempty(strfind(vinst, ' ')) == 0
275         blank = strfind(vinst, ' ');
276         vinst(blank) = [];
277     end
278     horse(i).vinst = [horse(i).vinst;
        str2double(vinst)];
279 end
280
281 if isnan(horse(i).odds) == 1
282     horse(i).loppid(end) = [];
283     horse(i).datum(end) = [];
284     horse(i).startnummer(end) = [];
285     horse(i).loppnummer(end) = [];
286     horse(i).distans(end) = [];
287     horse(i).tid(end) = [];
288     horse(i).resultat(end) = [];
289     horse(i).ntungbana(end) = [];
290     horse(i).odds(end) = [];
291     horse(i).kusk(end) = [];
292     horse(i).trainer(end) = [];
293     horse(i).vinterbana(end) = [];
294     horse(i).tungbana(end) = [];

```

```

295         horse(i).autostart(end) = [];
296         horse(i).galopp(end) = [];
297         horse(i).skorfram(end) = [];
298         horse(i).skorbak(end) = [];
299         horse(i).vinst(end) = [];
300         horse(i).bana(end) = [];
301     end
302 end
303 end
304
305     if isempty(horse(i).loppid) == 0
306         horse_id = [horse_id; n];
307         horse(i).id = n;
308         i = i + 1;
309     else
310         horse(end) = [];
311     end
312     fclose(fid);
313 end
314 end
315
316 save('horse','horse');
317 toc

```

```

1  %% Creating a single struct with new variables from several
   structs
2
3  loppdatum=121012; %No races after this date will be
   included
4
5  load('longhorse') %Load file
6
7  namn = []; id = []; alder = []; segerform = []; Dtemp = [];
8  banatrivselindex = 1; distanstrivselindex = 1;
   autotrivselindex = 1;
9  sommartrivselindex = 1;
10
11 redhorse = horse2redhorse(horse,loppdatum); %Function
12
13 for j =1:length(redhorse)
14     id = [id; redhorse(j).id];
15 end
16
17 antal = length(id);
18
19 birthdate = zeros(antal,1); ras = zeros(antal,1); loppid =
   zeros(antal,1); datum = zeros(antal,1);
20 startnummer = zeros(antal,1); loppnummer = zeros(antal,1);
   distans=zeros(antal,1);

```

```

21 tid = zeros(antal,1);resultat=zeros(antal,1);ntungbana =
    zeros(antal,1);odds = zeros(antal,1);
22 vinterbana = zeros(antal,1);
23 tungbana = zeros(antal,1); autostart = zeros(antal,1);
    galopp = zeros(antal,1);
24 skorfram = zeros(antal,1);skorbak = zeros(antal,1);vinst =
    zeros(antal,1);
25 kusk = {zeros(antal,1)}; trainer = {zeros(antal,1)}; bana =
    {zeros(antal,1)};
26 gender = {zeros(antal,1)};
27
28 start = 1;startpindex = 1;vinstformindex = 1;segerformindex
    =1;
29
30 vinstform = zeros(antal,1);
31 startp = zeros(antal,1);
32 trivselfaktor1 = zeros(antal,1);
33 banavinsttrivsel = zeros(antal,1);
34 banatopp3trivsel = zeros(antal,1);
35 distansvinsttrivsel = zeros(antal,1);
36 distanstopp3trivsel = zeros(antal,1);
37 autovinsttrivsel = zeros(antal,1);
38 autotopp3trivsel = zeros(antal,1);
39 sommarvinsttrivsel = zeros(antal,1);
40 sommartopp3trivsel = zeros(antal,1);
41
42 for j=1:length(redhorse)
43     stop = start + length(redhorse(j).id) - 1;
44     namn = [namn; {redhorse(j).namn}];
45     resultat(start:stop) = redhorse(j).resultat;
46     startnummer(start:stop) = redhorse(j).startnummer;
47     loppnummer(start:stop) = redhorse(j).loppnummer;
48     distans(start:stop) = redhorse(j).distans;
49     tid(start:stop) = redhorse(j).tid;
50     ntungbana(start:stop) = redhorse(j).ntungbana;
51     vinterbana(start:stop) = redhorse(j).vinterbana;
52     tungbana(start:stop) = redhorse(j).tungbana;
53     odds(start:stop) = redhorse(j).odds;
54     kusk(start:stop) = redhorse(j).kusk;
55     trainer(start:stop) = redhorse(j).trainer;
56     autostart(start:stop) = redhorse(j).autostart;
57     galopp(start:stop) = redhorse(j).galopp;
58     skorfram(start:stop) = redhorse(j).skorfram;
59     skorbak(start:stop) = redhorse(j).skorbak;
60     vinst(start:stop) = redhorse(j).vinst;
61     bana(start:stop) = redhorse(j).bana;
62     datum(start:stop) = redhorse(j).datum;
63     loppid(start:stop) = redhorse(j).loppid;
64     for r = start:stop

```

```

65         gender(r) = {redhorse(j).gender};
66     end
67
68     start = stop + 1;
69
70     % Age
71     fodd = num2str(horse(j).birthdate);
72     alder = [alder; round(redhorse(j).datum/10^4) - round(
73         str2double(fodd(3:8))/10^4)];
74
75     % Money shape
76     [vinstform, vinstformindex] = vinstformfunc(redhorse(j)
77         ,vinstform,vinstformindex);
78
79     % Win shape
80     [segerform, segerformindex] = segerformfunc(redhorse(j)
81         ,segerform,segerformindex);
82
83     % Starting point
84     [startp, startpindex] = startpfunc(redhorse(j),startp,
85         startpindex);
86
87     % Distance/track/starting/season fit
88     won = find(redhorse(j).resultat == 1);
89     second = find(redhorse(j).resultat == 2);
90     third = find(redhorse(j).resultat == 3);
91     topp3 = [won;second;third];
92     if isempty(won) == 0
93         vinstprocent = length(won)/length(redhorse(j).
94             resultat);
95     else
96         vinstprocent = 0;
97     end
98     if isempty(toppp3) == 0
99         topp3procent = length(toppp3)/length(redhorse(j).
100             resultat);
101     else
102         topp3procent = 0;
103     end
104     if isempty(redhorse(j).resultat) == 0
105
106         % Track fit
107         [banavinsttrivsel, banatopp3trivsel,
108             banatrivselindex] = trivselbana(banavinsttrivsel,
109                 banatopp3trivsel, banatrivselindex,...
110                 redhorse(j), topp3, won, topp3procent, vinstprocent
111                 );

```

```

105      % Distance fit
106      [distanvinsttrivsel, distanstopp3trivsel,
        distanstnivselindex] = trivseldistans(
        distansvinsttrivsel, distanstopp3trivsel,
        distanstnivselindex,...
107      redhorse(j), topp3, won, topp3procent, vinstprocent
        );
108
109      % Starting fit
110      [autovinsttrivsel, autotopp3trivsel,
        autotnivselindex] = trivselauto(
        autovinsttrivsel, autotopp3trivsel,
        autotnivselindex,...
111      redhorse(j), topp3, won, topp3procent, vinstprocent
        );
112
113      % Season fit
114      [sommavinsttrivsel, sommartopp3trivsel,
        sommartrivselindex] = trivselsommar(
        sommarvinsttrivsel, sommartopp3trivsel,
        sommartrivselindex,...
115      redhorse(j), topp3, won, topp3procent, vinstprocent
        );
116      end
117
118
119  end
120
121  % Driver rank
122  [kuskkrank, brakusk, ganskabarakusk, mycketbrakusk] =
        kuskkrankfunc(redhorse, resultat);
123
124  % Gender dummy
125  [valack, hingst, sto] = genderfunc(gender);
126
127  % Horse rank
128  [vinstprocent, platsprocent, mktbrahorse, brahorse,
        ganskabrahorse] =...
129  horserankfunc(redhorse, resultat);
130
131  % Money and time
132  [pengar, mintid] = pengar_mintid_func(redhorse, resultat);
133
134  % Dummy matrix
135  if j == 1
136      dummie = sparse([zeros(length(redhorse(j).resultat),
        length(redhorse)-1)]);
137  else

```

```

138     dummie = sparse([zeros(length(redhorse(j).resultat),j
      -2)...
139     ones(length(redhorse(j).resultat),1) zeros(length(
      redhorse(j).resultat),...
140     length(redhorse)-j)]);
141 end
142 Dtemp = [Dtemp; dummie];
143 etta = find(Dtemp == 1);
144 D = zeros(size(Dtemp));
145 D(etta) = 1;
146
147 superhorse.id = id;
148 superhorse.resultat = resultat;
149 superhorse.startnummer = startnummer;
150 superhorse.loppnummer = loppnummer;
151 superhorse.distans = distans;
152 superhorse.tid = tid;
153 superhorse.ntungbana = ntungbana;
154 superhorse.vinterbana = vinterbana;
155 superhorse.tungbana = tungbana;
156 superhorse.odds = odds;
157 superhorse.kusk = kusk;
158 superhorse.trainer = trainer;
159 superhorse.autostart = autostart;
160 superhorse.galopp = galopp;
161 superhorse.skorfram = skorfram;
162 superhorse.skorbak = skorbak;
163 superhorse.vinst = vinst;
164 superhorse.alder = alder;
165 superhorse.vinstform = vinstform;
166 superhorse.startp = startp;
167 superhorse.datum = datum;
168 superhorse.namn = namn;
169 superhorse.trivselfaktor = trivselfaktor1;
170 superhorse.brakusk = brakusk;
171 superhorse.ganskabrakusk = ganskabrakusk;
172 superhorse.kuskrank = kuskrank;
173 superhorse.banavinsttrivsel = banavinsttrivsel;
174 superhorse.banatopp3trivsel = banatopp3trivsel;
175 superhorse.distansvinsttrivsel = distansvinsttrivsel;
176 superhorse.distanstopp3trivsel = distanstopp3trivsel;
177 superhorse.autovinsttrivsel = autovinsttrivsel;
178 superhorse.autotopp3trivsel = autotopp3trivsel;
179 superhorse.sommarvinsttrivsel = sommarvinsttrivsel;
180 superhorse.sommartopp3trivsel = sommartopp3trivsel;
181 superhorse.valack = valack;
182 superhorse.hingst = hingst;
183 superhorse.sto = sto;
184 superhorse.mycketbrakusk = mycketbrakusk;

```

```

185 superhorse.mktbrahorse = mktbrahorse;
186 superhorse.brahorse = brahorse;
187 superhorse.ganskabrahorse = ganskabrahorse;
188 superhorse.vinstprocent = vinstprocent;
189 superhorse.pengar = pengar;
190 superhorse.loppid = loppid;
191 superhorse.platsprocent = platsprocent;
192 superhorse.segerform = segerform;
193 superhorse.mintid=mintid;
194
195 & Starting number rank, car start
196 RankA=tiedrank(-[0.1823 0.1910 0.2024 0.2261 0.2386 0.1877
    0.1583 0.1312 0.1288...
197 0.1369 0.1542 0.1172 0.1538 0.1176 0.1400]);
198
199 % Starting number rank, volt start
200 RankV=tiedrank(-[0.2289 0.2036 0.1963 0.1528 0.1653 0.2151
    0.1991 0.1510 0.1472...
201 0.1362 0.1426 0.1214]);
202
203 StartnummerRank=[];
204 for n=1:length(superhorse.startnummer)
205     if isnan(superhorse.startnummer(n))==1
206         superhorse.startnummer(n)=randi([1, 12]);
207     end
208     if superhorse.autostart(n) == 1;
209         StartnummerRank=[StartnummerRank; RankA(superhorse.
            startnummer(n))];
210     else
211         StartnummerRank=[StartnummerRank; RankV(superhorse.
            startnummer(n))];
212     end
213 end
214
215 n=1;
216 % Placement
217 omvresultat = zeros(length(superhorse.resultat),1);
218 for n=1:length(superhorse.resultat)
219     if superhorse.resultat(n)==0
220         omvresultat(n)=10;
221     else
222         omvresultat(n)=superhorse.resultat(n);
223     end
224 end
225
226 superhorse.StartnummerRank = StartnummerRank;
227 superhorse.omvresultat = omvresultat;

```

```

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

```

```

2
3 valack = zeros(length(gender), 1);
4 hingst = zeros(length(gender), 1);
5 sto = zeros(length(gender), 1);
6
7 index = 1;
8 for j = 1:length(gender)
9     if strcmp(gender(j), 'valack') == 1
10         valack(index) = 1;
11     elseif strcmp(gender(j), 'hingst') == 1
12         hingst(index) = 1;
13     elseif strcmp(gender(j), 'sto') == 1
14         sto(index) = 1;
15     end
16     index = index+1;
17 end

```

```

1 function redhorse = horse2redhorse(horse,loppdatum)
2 redhorse = struct;
3 redhorse.namn = [];
4 redhorse.id = [];
5 redhorse.color = [];
6 redhorse.gender = [];
7 redhorse.birthdate = [];redhorse.ras = [];redhorse.loppid =
8     [];redhorse.datum = [];
9 redhorse.startnummer = [];redhorse.loppnummer = [];redhorse
10     .distans=[];
11 redhorse.tid = [];redhorse.resultat=[];redhorse.ntungbana =
12     [];redhorse.odds = [];
13 redhorse.kusk = [];redhorse.trainer = [];redhorse.
14     vinterbana = [];
15 redhorse.tungbana = [];redhorse.autostart = [];redhorse.
16     galopp = [];
17 redhorse.skorfram = [];redhorse.skorbak = [];redhorse.vinst
18     = []; redhorse.bana = [];
19
20 s=0;
21 for j=1:length(horse)
22     redhorse(j).namn = horse(j).namn;
23     k = 1;
24     while k <= length(horse(j).datum)
25         if horse(j).datum(k) < loppdatum
26             for i =k:length(horse(j).datum)
27                 redhorse(j).id = [redhorse(j).id; horse(j).
28                     id];
29             end
30         end
31         redhorse(j).resultat = [redhorse(j).resultat; horse
32             (j).resultat(k:end)];
33     end
34 end

```



```

25     redhorse(j).startnummer = [redhorse(j).startnummer;
26         horse(j).startnummer(k:end)];
27     redhorse(j).loppnummer = [redhorse(j).loppnummer;
28         horse(j).loppnummer(k:end)];
29     redhorse(j).distans = [redhorse(j).distans; horse(j)
30         ).distans(k:end)];
31     redhorse(j).tid = [redhorse(j).tid; horse(j).tid(k:
32         end)];
33     redhorse(j).ntungbana = [redhorse(j).ntungbana;
34         horse(j).ntungbana(k:end)];
35     redhorse(j).vinterbana = [redhorse(j).vinterbana;
36         horse(j).vinterbana(k:end)];
37     redhorse(j).tungbana = [redhorse(j).tungbana; horse
38         (j).tungbana(k:end)];
39     redhorse(j).odds = [redhorse(j).odds; horse(j).odds
40         (k:end)];
41     redhorse(j).kusk = [redhorse(j).kusk; horse(j).kusk
42         (k:end)];
43     redhorse(j).trainer = [redhorse(j).trainer; horse(j)
44         ).trainer(k:end)];
45     redhorse(j).autostart = [redhorse(j).autostart;
46         horse(j).autostart(k:end)];
47     redhorse(j).galopp = [redhorse(j).galopp; horse(j).
48         galopp(k:end)];
49     redhorse(j).skorfram = [redhorse(j).skorfram; horse
50         (j).skorfram(k:end)];
51     redhorse(j).skorbak = [redhorse(j).skorbak; horse(j)
52         ).skorbak(k:end)];
53     redhorse(j).vinst = [redhorse(j).vinst; horse(j).
54         vinst(k:end)];
55     redhorse(j).datum = [redhorse(j).datum; horse(j).
56         datum(k:end)];
57     redhorse(j).bana = [redhorse(j).bana; horse(j).bana
58         (k:end)];
59     redhorse(j).gender = horse(j).gender;
60     redhorse(j).loppid = [redhorse(j).loppid; horse(j).
61         loppid(k:end)];
62
63     nanvektor = find(isnan(redhorse(j).odds));
64     if isempty(nanvektor) == 0
65         nanvektor = fliplr(nanvektor');
66         for m = 1:length(nanvektor')
67             s=s+1;
68             n = nanvektor(m);
69
70             redhorse(j).id(end) = [];
71             redhorse(j).resultat(n) = [];
72             redhorse(j).startnummer(n) = [];
73             redhorse(j).loppnummer(n) = [];

```

```

56         redhorse(j).distans(n) = [];
57         redhorse(j).tid(n) = [];
58         redhorse(j).ntungbana(n) = [];
59         redhorse(j).vinterbana(n) = [];
60         redhorse(j).tungbana(n) = [];
61         redhorse(j).odds(n) = [];
62         redhorse(j).kusk(n) = [];
63         redhorse(j).trainer(n) = [];
64         redhorse(j).autostart(n) = [];
65         redhorse(j).galopp(n) = [];
66         redhorse(j).skorfram(n) = [];
67         redhorse(j).skorbak(n) = [];
68         redhorse(j).vinst(n) = [];
69         redhorse(j).datum(n) = [];
70         redhorse(j).bana(n) = [];
71         redhorse(j).loppid(n) = [];
72     end
73     end
74     k = length(horse(j).datum);
75 end
76 k = k+1;
77 end
78 end
79 end

```

```

1 function [vinstprocent, platsprocent, mktbrahorse, brahorse
, ganskabrahorse] =...
2 horserankfunc(redhorse,resultat)
3
4 vinstprocent = zeros(length(resultat),1);
5 platsprocent = zeros(length(resultat),1);
6 antalloppvec = zeros(length(resultat),1);
7 mktbrahorse = zeros(length(resultat),1);
8 brahorse = zeros(length(resultat),1);
9 ganskabrahorse = zeros(length(resultat),1);
10 pengar = zeros(length(resultat),1);
11 idhorse = zeros(length(resultat),1);
12 mintid = zeros(length(resultat),1);
13
14 n=1;
15 for j=1:length(redhorse)
16     antallopptot = length(redhorse(j).resultat);
17     for i=2:antallopptot
18         distans1 = redhorse(j).distans(i-1);
19         index = find(redhorse(j).distans(i:end)==distans1);
20         if isempty(index) == 1
21             mintid(n) = 0;
22         else
23             mintid(n) = min(redhorse(j).tid(index));

```

```

24         if isnan(mintid(n))==1
25             mintid(n) = 0;
26         end
27     end
28
29     antallopp = length(redhorse(j).resultat(i:end));
30     antalvinster = length(find(redhorse(j).resultat(i:
31         end) == 1));
32     antalplatser = antalvinster + length(find(redhorse(
33         j).resultat(i:end) == 2)) + length(find(
34         redhorse(j).resultat(i:end) == 3));
35     pengar(n) = sum(redhorse(j).vinst(i:end))/antallopp
36     ;
37     vinstprocent(n) = antalvinster/antallopp;
38     platsprocent(n) = antalplatser/antallopp;
39     antalloppvec(n) = antallopp;
40
41     idhorse(n) = redhorse(j).id(i);
42     if vinstprocent(n) >= 0.3 && antallopp > 6
43         mktbrahorse(n)=1;
44     elseif vinstprocent(n) >= 0.20
45         brahorse(n)=1;
46     elseif vinstprocent(n) >= 0.15
47         ganskabrahorse(n)=1;
48     end
49     n = n+1;
50 end
51 end

```

```

1 function [kuskrank, brakusk, ganskabrakusk, mycketbrakusk]
2     = kuskrankfunc(redhorse, resultat)
3
4 load('kuskar.mat');
5 load('kuskprocent');
6
7 kuskrank = zeros(length(resultat),1);
8 brakusk = zeros(length(resultat),1);
9 ganskabrakusk = zeros(length(resultat),1);
10 mycketbrakusk = zeros(length(resultat),1);
11
12 n=1;
13 for j=1:length(redhorse)
14     for i=1:length(redhorse(j).resultat)
15         plats=find(strcmp(kuskar,redhorse(j).kusk(i)) == 1)
16         ;

```

```

15         if isempty(plats)==1
16             mycketbrakusk(n)=0;
17             brakusk(n)=0;
18             ganskabrakusk(n)=0;
19             n=n+1;
20         else
21             kuskrank(n)=kuskprocent(plats);
22             if kuskprocent(plats)>0.2 && kuskprocent(plats)
                <0.32
23                 mycketbrakusk(n)=1;
24                 brakusk(n)=0;
25                 ganskabrakusk(n)=0;
26                 n=n+1;
27             elseif kuskprocent(plats)>0.16 && kuskprocent(
                plats)<=0.2
28                 mycketbrakusk(n)=0;
29                 brakusk(n)=1;
30                 ganskabrakusk(n)=0;
31                 n=n+1;
32             elseif kuskprocent(plats)>0.13 && kuskprocent(
                plats)<=0.16
33                 mycketbrakusk(n)=0;
34                 brakusk(n)=0;
35                 ganskabrakusk(n)=1;
36                 n=n+1;
37             else
38                 mycketbrakusk(n)=0;
39                 brakusk(n)=0;
40                 ganskabrakusk(n)=0;
41                 n=n+1;
42             end
43         end
44     end
45 end

```

```

1 function [pengar, mintid] = pengar_mintid_func(redhorse,
    resultat)
2
3 pengar = zeros(length(resultat),1);
4 mintid = zeros(length(resultat),1);
5
6 n=1;
7 for j=1:length(redhorse)
8     antallopptot = length(redhorse(j).resultat);
9     for i=2:antallopptot
10         if redhorse(j).tid(i)<=9
11             redhorse(j).tid(i)=NaN;
12         end
13     end

```

```

14
15     for i=2:antallopptot
16         k=redhorse(j).autostart(i-1);
17         antallopp = length(redhorse(j).resultat(i:end));
18         distans1 = redhorse(j).distans(i-1);
19         index = find(redhorse(j).distans(i:end)>=(distans1
20             -60) & redhorse(j).autostart(i:end)==k &
21             redhorse(j).distans(i:end)<=(distans1+60));
22         index = index + i - 1;
23
24         if isempty(index) == 1
25             mintid(n) = 0;
26         else
27             mintid(n) = min(redhorse(j).tid(index));
28             if isnan(mintid(n))==1
29                 mintid(n) = 0;
30             end
31         end
32
33         pengaar(n) = sum(redhorse(j).vinst(i:end))/antallopp;
34         n = n+1;
35     end
36
37     if antallopptot > 0
38         n = n+1;
39     end
40 end

```

```

1 function [startp, startpindex] = startpfunc(redhorse,startp
2     , startpindex)
3
4 startstartp = 1;
5 stopstartp = length(redhorse.vinst);
6
7 if length(redhorse.vinst) > 5
8     startstartp = 1;
9     stopstartp = length(redhorse.vinst);
10    for k = startstartp:stopstartp-5
11        startp_vinst = 0;
12        for n = k + 1:k + 5
13            if redhorse.resultat(n) == 1
14                startp_vinst = startp_vinst + 300;
15            elseif redhorse.resultat(n) == 2
16                startp_vinst = startp_vinst + 150;
17            elseif redhorse.resultat(n) == 3
18                startp_vinst = startp_vinst + 100;
19            elseif redhorse.resultat(n) == 4
20                startp_vinst = startp_vinst + 50;
21            elseif redhorse.resultat(n) == 5

```

```

21         startp_vinst = startp_vinst + 25;
22     end
23 end
24
25     startp(startpindex) = sum(redhorse.vinst(k+1:k+5))
26         /100 + startp_vinst;
27     startpindex = startpindex + 1;
28 end
29
30 for m =2:5
31     startp_vinst = 0;
32     for n = k + m:k + 5
33         if redhorse.resultat(n) == 1
34             startp_vinst = startp_vinst + 300;
35         elseif redhorse.resultat(n) == 2
36             startp_vinst = startp_vinst + 150;
37         elseif redhorse.resultat(n) == 3
38             startp_vinst = startp_vinst + 100;
39         elseif redhorse.resultat(n) == 4
40             startp_vinst = startp_vinst + 50;
41         elseif redhorse.resultat(n) == 5
42             startp_vinst = startp_vinst + 25;
43         end
44     end
45
46     startp(startpindex) = sum(redhorse.vinst(k+m:k+5))
47         /100 + startp_vinst;
48     startpindex = startpindex + 1;
49 end
50
51 startp(startpindex) = 0;
52 startpindex = startpindex + 1;
53
54 elseif length(redhorse.vinst) == 5
55     for m =1:4
56         startp_vinst = 0;
57
58         for n = startstartp + m:startstartp + 4
59             if redhorse.resultat(n) == 1
60                 startp_vinst = startp_vinst + 300;
61             elseif redhorse.resultat(n) == 2
62                 startp_vinst = startp_vinst + 150;
63             elseif redhorse.resultat(n) == 3
64                 startp_vinst = startp_vinst + 100;
65             elseif redhorse.resultat(n) == 4
66                 startp_vinst = startp_vinst + 50;
67             elseif redhorse.resultat(n) == 5
68                 startp_vinst = startp_vinst + 25;
69             end

```

```

68         end
69
70         startp(startpindex) = sum(redhorse.vinst(
71             startstartp+m:startstartp+4))/100 + startp_vinst;
72         startpindex = startpindex + 1;
73     end
74
75     startp(startpindex) = 0;
76     startpindex = startpindex + 1;
77 elseif length(redhorse.vinst) == 4
78     for m =1:3
79         startp_vinst = 0;
80         for n = startstartp + m:startstartp + 3
81             if redhorse.resultat(n) == 1
82                 startp_vinst = startp_vinst + 300;
83             elseif redhorse.resultat(n) == 2
84                 startp_vinst = startp_vinst + 150;
85             elseif redhorse.resultat(n) == 3
86                 startp_vinst = startp_vinst + 100;
87             elseif redhorse.resultat(n) == 4
88                 startp_vinst = startp_vinst + 50;
89             elseif redhorse.resultat(n) == 5
90                 startp_vinst = startp_vinst + 25;
91             end
92         end
93
94         startp(startpindex) = sum(redhorse.vinst(
95             startstartp+m:startstartp+3))/100 +
96         startp_vinst;
97         startpindex = startpindex + 1;
98     end
99
100     startp(startpindex) = 0;
101     startpindex = startpindex + 1;
102
103 elseif length(redhorse.vinst) == 3
104     for m =1:2
105         startp_vinst = 0;
106         for n = startstartp + m:startstartp + 2
107             if redhorse.resultat(n) == 1
108                 startp_vinst = startp_vinst + 300;
109             elseif redhorse.resultat(n) == 2
110                 startp_vinst = startp_vinst + 150;
111             elseif redhorse.resultat(n) == 3
112                 startp_vinst = startp_vinst + 100;
113             elseif redhorse.resultat(n) == 4
114                 startp_vinst = startp_vinst + 50;
115             elseif redhorse.resultat(n) == 5

```

```

114         startp_vinst = startp_vinst + 25;
115     end
116 end
117
118     startp(startpindex) = sum(redhorse.vinst(
119         startstartp+m:startstartp+2))/100 + startp_vinst;
120     startpindex = startpindex + 1;
121 end
122 startp(startpindex) = 0;
123 startpindex = startpindex + 1;
124
125 elseif length(redhorse.vinst) == 2
126     startp_vinst = 0;
127     for n = startstartp + 1:startstartp + 1
128         if redhorse.resultat(n) == 1
129             startp_vinst = startp_vinst + 300;
130         elseif redhorse.resultat(n) == 2
131             startp_vinst = startp_vinst + 150;
132         elseif redhorse.resultat(n) == 3
133             startp_vinst = startp_vinst + 100;
134         elseif redhorse.resultat(n) == 4
135             startp_vinst = startp_vinst + 50;
136         elseif redhorse.resultat(n) == 5
137             startp_vinst = startp_vinst + 25;
138         end
139     end
140
141     startp(startpindex) = sum(redhorse.vinst(startstartp+1:
142         startstartp+1))/100 + startp_vinst;
143     startpindex = startpindex + 1;
144     startp(startpindex) = 0;
145     startpindex = startpindex + 1;
146
147 elseif length(redhorse.vinst) == 1
148     startp(startpindex) = 0;
149     startpindex = startpindex + 1;
150 end

```

```

1 function [segerform, segerformindex] = segerformfunc(
2     redhorse,segerform, segerformindex)
3
4 tempres=redhorse.resultat;
5
6 for n=1:length(redhorse.resultat)
7     if tempres(n)==0
8         tempres(n)=10;
9     end
10 end
11
12 if length(redhorse.vinst) > 3

```



```

11     for k = 1:length(redhorse.vinst)-3
12         segerform(segerformindex) = sum(tempres(k+1:k+3))
           /3;
13         segerformindex = segerformindex+1;
14     end
15
16     segerform(segerformindex) = sum(tempres(k+2:k+3))/2;
17     segerform(segerformindex + 1) = sum(tempres(k+3:k+3));
18     segerform(segerformindex + 2) = 5;
19     segerformindex = segerformindex+3;
20
21 elseif length(redhorse.vinst) == 3
22     segerform(segerformindex) = sum(tempres(2:3))/2;
23     segerform(segerformindex + 1) = sum(tempres(3:3));
24     segerform(segerformindex + 2) = 5;
25     segerformindex = segerformindex+3;
26
27 elseif length(redhorse.vinst) == 2
28     segerform(segerformindex) = sum(tempres(2:2));
29     segerform(segerformindex + 1) = 5;
30     segerformindex = segerformindex+2;
31
32 elseif length(redhorse.vinst) == 1
33     segerform(segerformindex) = 5;
34     segerformindex = segerformindex+1;
35
36 else
37     end

```

```

1 function [vinstform, vinstformindex] = vinstformfunc(
   redhorse, vinstform, vinstformindex)
2
3 if length(redhorse.vinst) > 3
4     for k = 1:length(redhorse.vinst)-3
5         vinstform(vinstformindex) = sum(redhorse.vinst(k+1:
           k+3))/1000;
6         vinstformindex = vinstformindex+1;
7     end
8
9     vinstform(vinstformindex) = sum(redhorse.vinst(k+2:k+3)
   )/1000;
10    vinstform(vinstformindex + 1) = sum(redhorse.vinst(k+3:
   k+3))/1000;
11    vinstform(vinstformindex + 2) = 0;
12    vinstformindex = vinstformindex+3;
13
14 elseif length(redhorse.vinst) == 3
15     vinstform(vinstformindex) = sum(redhorse.vinst(2:3))
   /1000;

```

```

16     vinstform(vinstformindex + 1) = sum(redhorse.vinst(3:3)
17         )/1000;
18     vinstform(vinstformindex + 2) = 0;
19     vinstformindex = vinstformindex+3;
20 elseif length(redhorse.vinst) == 2
21     vinstform(vinstformindex) = sum(redhorse.vinst(2:2))
22         /1000;
23     vinstform(vinstformindex + 1) = 0;
24     vinstformindex = vinstformindex+2;
25 elseif length(redhorse.vinst) == 1
26     vinstform(vinstformindex) = 0;
27     vinstformindex = vinstformindex+1;
28 else
29 end

```

```

1 function [gvek, hvek, autotrivselindex] = trivselauto(gvek,
2     hvek, autotrivselindex, redhorse, topp3, won, topp3procent
3     , vinstprocent)
4
5 if length(redhorse.resultat) > 5
6     vinstvec = zeros(length(redhorse.resultat),1);
7     topp3vec = zeros(length(redhorse.resultat),1);
8     resultat = redhorse.resultat;
9
10    vinstprocentvec = zeros(length(redhorse.resultat),1);
11    topp3procentvec = zeros(length(redhorse.resultat),1);
12    vinstprocentavec=zeros(length(redhorse.resultat),1);
13    topp3procentavec=zeros(length(redhorse.resultat),1);
14    vinstprocentvvec=zeros(length(redhorse.resultat),1);
15    topp3procentvvec=zeros(length(redhorse.resultat),1);
16
17    index = find(resultat==1);
18    vinstvec(index)=1;
19    index = find(resultat >0 & resultat <4);
20    topp3vec(index)=1;
21    avec=redhorse.autostart;
22    vvec = ones(length(redhorse.resultat),1)-avec;
23
24    vinstavec=vinstvec.*avec;
25    vinstvvec=vinstvec.*vvec;
26    topp3avec=topp3vec.*avec;
27    topp3vvec=topp3vec.*vvec;
28
29    for k = 1:length(redhorse.resultat)
30        vinstprocentvec(k)=sum(vinstvec(k:end))/length(
31            vinstvec(k:end));

```

```

29     topp3procentvec(k)=sum(top3vec(k:end))/length(
30         topp3vec(k:end));
31
32     if sum(avec(k:end)) >0
33         vinstprocentavec(k)=sum(vinstavec(k:end))/sum(
34             avec(k:end));
35         topp3procentavec(k)=sum(top3avec(k:end))/sum(
36             avec(k:end));
37     end
38
39     if sum(vvec(k:end)) >0
40         vinstprocentvvec(k)=sum(vinstvvec(k:end))/sum(
41             vvec(k:end));
42         topp3procentvvec(k)=sum(top3vvec(k:end))/sum(
43             vvec(k:end));
44     end
45
46     end
47
48     for k = 1:length(redhorse.resultat)-5
49         n = k+1;
50
51         if avec(k)==1 && vinstprocentavec(n)/
52             vinstprocentvec(n) > 1.05
53             gvek(autotrivselindex) = 1;
54         end
55
56         if avec(k)==1 && topp3procentavec(n)/
57             topp3procentvec(n) > 1.05
58             hvek(autotrivselindex) = 1;
59         end
60
61         if vvec(k)==1 && vinstprocentvvec(n)/
62             vinstprocentvec(n) > 1.05
63             gvek(autotrivselindex) = 1;
64         end
65
66         if vvec(k)==1 && topp3procentvvec(n)/
67             topp3procentvec(n) > 1.05
68             hvek(autotrivselindex) = 1;
69         end
70
71         autotrivselindex=autotrivselindex+1;
72     end
73
74     for k = 1:5
75         gvek(autotrivselindex) = 0;
76         hvek(autotrivselindex) = 0;
77         autotrivselindex = autotrivselindex + 1;
78     end
79
80     else

```

```

69     for k = 1:length(redhorse.resultat)
70         gvek(autotrivselindex) = 0;
71         hvek(autotrivselindex) = 0;
72         autotrivselindex = autotrivselindex + 1;
73     end
74 end

1 function [evek, fvek, distanstriivselindex] = trivseldistans
   (evek,fvek, distanstriivselindex, redhorse, topp3, won,
   topp3procent, vinstprocent)
2
3 if length(redhorse.resultat) > 5
4     vinstvec = zeros(length(redhorse.resultat),1);
5     topp3vec = zeros(length(redhorse.resultat),1);
6     kdistvec = zeros(length(redhorse.resultat),1);
7     mdistvec = zeros(length(redhorse.resultat),1);
8     ldistvec = zeros(length(redhorse.resultat),1);
9
10    resultat = redhorse.resultat;
11
12    vinstprocentvec = zeros(length(redhorse.resultat),1);
13    topp3procentvec = zeros(length(redhorse.resultat),1);
14    vinstprocentkdistvec=zeros(length(redhorse.resultat),1)
15    ;
16    topp3procentkdistvec=zeros(length(redhorse.resultat),1)
17    ;
18    vinstprocentmdistvec=zeros(length(redhorse.resultat),1)
19    ;
20    topp3procentmdistvec=zeros(length(redhorse.resultat),1)
21    ;
22    vinstprocentldistvec=zeros(length(redhorse.resultat),1)
23    ;
24    topp3procentldistvec=zeros(length(redhorse.resultat),1)
25    ;
26
27    for k = 1:length(resultat)
28        if resultat(k)==1
29            vinstvec(k)=1;
30        end
31        if resultat(k) < 4 && resultat(k) > 0
32            topp3vec(k)=1;
33        end
34        dist=redhorse.distans(k);
35
36        if isnan(dist)==1
37        elseif dist <1700
38            kdistvec(k) = 1;
39        elseif dist > 2400
40            ldistvec(k) = 1;

```

```

35         else
36             mdistvec(k) = 1;
37         end
38     end
39
40     vinstkdistvec=vinstvec.*kdistvec;
41     vinstmdistvec=vinstvec.*mdistvec;
42     vinstldistvec=vinstvec.*ldistvec;
43     topp3kdistvec=topp3vec.*kdistvec;
44     topp3mdistvec=topp3vec.*mdistvec;
45     topp3ldistvec=topp3vec.*ldistvec;
46
47     for k = 1:length(redhorse.resultat)
48         vinstprocentvec(k)=sum(vinstvec(k:end))/length(
49             vinstvec(k:end));
50         topp3procentvec(k)=sum(topp3vec(k:end))/length(
51             topp3vec(k:end));
52
53         if sum(kdistvec(k:end)) >0
54             vinstprocentkdistvec(k)=sum(vinstkdistvec(k:end)
55                 )/sum(kdistvec(k:end));
56             topp3procentkdistvec(k)=sum(topp3kdistvec(k:end)
57                 )/sum(kdistvec(k:end));
58         end
59
60         if sum(mdistvec(k:end)) >0
61             vinstprocentmdistvec(k)=sum(vinstmdistvec(k:end)
62                 )/sum(mdistvec(k:end));
63             topp3procentmdistvec(k)=sum(topp3mdistvec(k:end)
64                 )/sum(mdistvec(k:end));
65         end
66
67         if sum(ldistvec(k:end)) >0
68             vinstprocentldistvec(k)=sum(vinstldistvec(k:end)
69                 )/sum(ldistvec(k:end));
70             topp3procentldistvec(k)=sum(topp3ldistvec(k:end)
71                 )/sum(ldistvec(k:end));
72         end
73     end
74
75     for k = 1:length(redhorse.resultat)-5
76         n = k+1;
77         if kdistvec(k)==1 && vinstprocentkdistvec(n)/
78             vinstprocentvec(n) > 1.05
79             evek(distanstrivselindex) = 1;
80         end
81
82         if kdistvec(k)==1 && topp3procentkdistvec(n)/
83             topp3procentvec(n) > 1.05

```

```

74         fvek(distanstrivselindex) = 1;
75     end
76
77     if mdistvec(k)==1 && vinstprocentmdistvec(n)/
78         vinstprocentvec(n) > 1.05
79         evek(distanstrivselindex) = 1;
80     end
81
82     if mdistvec(k)==1 && topp3procentmdistvec(n)/
83         topp3procentvec(n) > 1.05
84         fvek(distanstrivselindex) = 1;
85     end
86
87     if ldistvec(k)==1 && vinstprocentldistvec(n)/
88         vinstprocentvec(n) > 1.05
89         evek(distanstrivselindex) = 1;
90     end
91
92     if ldistvec(k)==1 && topp3procentldistvec(n)/
93         topp3procentvec(n) > 1.05
94         fvek(distanstrivselindex) = 1;
95     end
96     distanstrivselindex=distanstrivselindex+1;
97 end
98
99 for k = 1:5
100     evek(distanstrivselindex) = 0;
101     fvek(distanstrivselindex) = 0;
102     distanstrivselindex = distanstrivselindex + 1;
103 end
104 else
105     for k = 1:length(redhorse.resultat)
106         evek(distanstrivselindex) = 0;
107         fvek(distanstrivselindex) = 0;
108         distanstrivselindex = distanstrivselindex + 1;
109     end
110 end
111 end

```

```

1 function [sommavinsttrivsel, sommartopp3trivsel,
2 sommartrivselindex] = trivsel_sommar(sommavinsttrivsel,
3 sommartopp3trivsel, sommartrivselindex, redhorse, topp3,
4 won, topp3procent, vinstprocent)
5
6 if length(redhorse.resultat) > 5
7     vinstvec = zeros(length(redhorse.resultat),1);
8     topp3vec = zeros(length(redhorse.resultat),1);
9     sommarvec = zeros(length(redhorse.resultat),1);
10    vinterver = zeros(length(redhorse.resultat),1);
11    resultat = redhorse.resultat;

```

```

9      vinstprocentvec = zeros(length(redhorse.resultat),1);
10     topp3procentvec = zeros(length(redhorse.resultat),1);
11     vinstprocentsommarvec=zeros(length(redhorse.resultat)
12     ,1);
12     topp3procentsommarvec=zeros(length(redhorse.resultat)
13     ,1);
13     vinstprocentvintervec=zeros(length(redhorse.resultat)
14     ,1);
14     topp3procentvintervec=zeros(length(redhorse.resultat)
15     ,1);
15
16     for k = 1:length(resultat)
17         if resultat(k)==1
18             vinstvec(k)=1;
19         end
20         if resultat(k) < 4 && resultat(k) > 0
21             topp3vec(k)=1;
22         end
23
24         a = redhorse.datum(k);
25         b = floor(a/100);
26         c = floor(b/100)*100;
27         month = b-c;
28
29         if month >= 4 && month <= 9
30             sommarvec(k) = 1;
31         else
32             vintervec(k) = 1;
33         end
34     end
35
36     vinstvintervec=vinstvec.*vintervec;
37     vinstsommarvec=vinstvec.*sommarvec;
38     topp3vintervec=topp3vec.*vintervec;
39     topp3sommarvec=topp3vec.*sommarvec;
40
41     for k = 1:length(redhorse.resultat)
42         vinstprocentvec(k)=sum(vinstvec(k:end))/length(
43         vinstvec(k:end));
43         topp3procentvec(k)=sum(topp3vec(k:end))/length(
44         topp3vec(k:end));
44
45         if sum(sommarvec(k:end)) >0
46             vinstprocentsommarvec(k)=sum(vinstsommarvec(k:
47             end))/sum(sommarvec(k:end));
47             topp3procentsommarvec(k)=sum(topp3sommarvec(k:
48             end))/sum(sommarvec(k:end));
48         end
49

```

```

50         if sum(vintervec(k:end)) > 0
51             vinstprocentvintervec(k)=sum(vinstvintervec(k:
52                 end))/sum(vintervec(k:end));
53             topp3procentvintervec(k)=sum(topp3vintervec(k:
54                 end))/sum(vintervec(k:end));
55         end
56     end
57
58     for k = 1:length(redhorse.resultat)-5
59         n = k+1;
60
61         if sommarvec(k)==1 && vinstprocentsommarvec(n)/
62             vinstprocentvec(n) > 1.05
63             sommarvinsttrivsel(sommartrivselindex) = 1;
64         end
65
66         if sommarvec(k)==1 && topp3procentsommarvec(n)/
67             topp3procentvec(n) > 1.05
68             sommartopp3trivsel(sommartrivselindex) = 1;
69         end
70
71         if vintervec(k)==1 && vinstprocentvintervec(n)/
72             vinstprocentvec(n) > 1.05
73             sommarvinsttrivsel(sommartrivselindex) = 1;
74         end
75
76         if vintervec(k)==1 && topp3procentvintervec(n)/
77             topp3procentvec(n) > 1.05
78             sommartopp3trivsel(sommartrivselindex) = 1;
79         end
80         sommartrivselindex=sommartrivselindex+1;
81     end
82
83     for k = 1:5
84         sommarvinsttrivsel(sommartrivselindex) = 0;
85         sommartopp3trivsel(sommartrivselindex) = 0;
86         sommartrivselindex = sommartrivselindex + 1;
87     end
88 else
89     for k = 1:length(redhorse.resultat)
90         sommarvinsttrivsel(sommartrivselindex) = 0;
91         sommartopp3trivsel(sommartrivselindex) = 0;
92         sommartrivselindex = sommartrivselindex + 1;
93     end
94 end

```

```

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

```



```

3 load superhorse % Load data set for model 1
4
5 load okloppid % Load id vector with whole races
6
7 minisuperhorse = struct;minisuperhorse.namn = [];
8 minisuperhorse.id = [];minisuperhorse.color = [];
9 minisuperhorse.gender = [];minisuperhorse.birthdate = [];
10 minisuperhorse.ras = [];minisuperhorse.loppid = [];
11 minisuperhorse.datum = [];
12 minisuperhorse.startnummer = [];minisuperhorse.loppnummer =
13 [];minisuperhorse.distans=[];
14 minisuperhorse.tid = [];minisuperhorse.resultat=[];
15 minisuperhorse.ntungbana = [];
16 minisuperhorse.odds = [];minisuperhorse.kusk = {};
17 minisuperhorse.trainer = [];minisuperhorse.vinterbana = [];
18 minisuperhorse.tungbana = [];minisuperhorse.autostart = [];
19 minisuperhorse.galopp = [];
20 minisuperhorse.skorfram = [];minisuperhorse.skorbak = [];
21 minisuperhorse.vinst = []; minisuperhorse.bana = [];
22 minisuperhorse.distansdummy = []; minisuperhorse.pengar =
23 []; minisuperhorse.alder = [];
24 minisuperhorse.trivselfaktor = []; minisuperhorse.brakusk =
25 []; minisuperhorse.mycketbrakusk = [];
26 minisuperhorse.ganskabrakusk = []; minisuperhorse.kuskrank
27 = []; minisuperhorse.valack = [];
28 minisuperhorse.sto = []; minisuperhorse.hingst = [];
29 minisuperhorse.mktbrahorse = [];
30 minisuperhorse.ganskabrahorse = []; minisuperhorse.brahorse
31 = []; minisuperhorse.vinstprocent = [];
32 minisuperhorse.platsprocent = []; minisuperhorse.startp =
33 []; minisuperhorse.vinstform = [];
34 minisuperhorse.segerform = [];minisuperhorse.
35 banavinsttrivsel=[];minisuperhorse.banatopp3trivsel=[];
36 minisuperhorse.distansvinsttrivsel=[];minisuperhorse.
37 distanstopp3trivsel=[];
38 minisuperhorse.autovinsttrivsel=[];minisuperhorse.
39 autotopp3trivsel=[];
40 minisuperhorse.sommarvinsttrivsel=[];minisuperhorse.
41 sommartopp3trivsel=[];
42 minisuperhorse.mintid = [];
43
44 for j=okloppid'
45     redhorse = struct;
46     redhorse.namn = [];
47     redhorse.id = [];
48     redhorse.color = [];
49     redhorse.gender = [];
50     redhorse.birthdate = [];redhorse.ras = [];redhorse.
51     loppid = [];redhorse.datum = [];

```

```

36 redhorse.startnummer = [];redhorse.loppnummer = [];
   redhorse.distans=[];
37 redhorse.tid = [];redhorse.resultat=[];redhorse.
   ntungbana = [];redhorse.odds = [];
38 redhorse.kusk = [];redhorse.trainer = [];redhorse.
   vinterbana = [];
39 redhorse.tungbana = [];redhorse.autostart = [];redhorse
   .galopp = [];
40 redhorse.skorfram = [];redhorse.skorbak = [];redhorse.
   vinst = []; redhorse.bana = [];
41 redhorse.distansdummy = [];
42
43 indexvec = find(superhorse.loppid==j);
44 redhorse.namn = superhorse.namn(indexvec);
45 redhorse.loppid = superhorse.loppid(indexvec);
46 redhorse.id = superhorse.id(indexvec);
47 redhorse.resultat = superhorse.resultat(indexvec);
48 redhorse.startnummer = superhorse.startnummer(indexvec)
   ;
49 redhorse.loppnummer = superhorse.loppnummer(indexvec);
50 redhorse.distans = superhorse.distans(indexvec);
51 redhorse.tid = superhorse.tid(indexvec);
52 redhorse.ntungbana = superhorse.ntungbana(indexvec);
53 redhorse.vinterbana = superhorse.vinterbana(indexvec);
54 redhorse.tungbana = superhorse.tungbana(indexvec);
55 redhorse.odds = superhorse.odds(indexvec);
56 redhorse.kusk = superhorse.kusk(indexvec);
57 redhorse.trainer = superhorse.trainer(indexvec);
58 redhorse.autostart = superhorse.autostart(indexvec);
59 redhorse.galopp = superhorse.galopp(indexvec);
60 redhorse.skorfram = superhorse.skorfram(indexvec);
61 redhorse.skorbak = superhorse.skorbak(indexvec);
62 redhorse.vinst = superhorse.vinst(indexvec);
63 redhorse.alder = superhorse.alder(indexvec);
64 redhorse.vinstform = superhorse.vinstform(indexvec);
65 redhorse.startp = superhorse.startp(indexvec);
66 redhorse.datum = superhorse.datum(indexvec);
67 redhorse.trivselfaktor = superhorse.trivselfaktor(
   indexvec);
68 redhorse.brakusk = superhorse.brakusk(indexvec);
69 redhorse.ganskabrukusk = superhorse.ganskabrukusk(
   indexvec);
70 redhorse.kuskrank = superhorse.kuskrank(indexvec);
71 redhorse.valack = superhorse.valack(indexvec);
72 redhorse.hingst = superhorse.hingst(indexvec);
73 redhorse.sto = superhorse.sto(indexvec);
74 redhorse.mycketbrukusk = superhorse.mycketbrukusk(
   indexvec);

```

```

75     redhorse.mktbrahorse = superhorse.mktbrahorse(indexvec)
76     ;
77     redhorse.brahorse = superhorse.brahorse(indexvec);
78     redhorse.ganskabrahorse = superhorse.ganskabrahorse(
79         indexvec);
80     redhorse.vinstprocent = superhorse.vinstprocent(
81         indexvec);
82     redhorse.pengar = superhorse.pengar(indexvec);
83     redhorse.platsprocent = superhorse.platsprocent(
84         indexvec);
85     redhorse.segerform = superhorse.segerform(indexvec);
86     redhorse.mintid = superhorse.mintid(indexvec);
87     redhorse.banavinsttrivsel=superhorse.banavinsttrivsel(
88         indexvec);
89     redhorse.banatopp3trivsel=superhorse.banatopp3trivsel(
90         indexvec);
91     redhorse.distansvinsttrivsel=superhorse.
92         distansvinsttrivsel(indexvec);
93     redhorse.distanstopp3trivsel=superhorse.
94         distanstopp3trivsel(indexvec);
95     redhorse.autovinsttrivsel=superhorse.autovinsttrivsel(
96         indexvec);
97     redhorse.autotopp3trivsel=superhorse.autotopp3trivsel(
98         indexvec);
99     redhorse.sommarvinsttrivsel=superhorse.
100         sommarvinsttrivsel(indexvec);
101     redhorse.sommartopp3trivsel=superhorse.
102         sommartopp3trivsel(indexvec);
103
104     mintid2=[];
105     for k=1:length(redhorse.resultat)
106         if redhorse.mintid(k) < 9
107             redhorse.mintid(k) = 0;
108         elseif redhorse.mintid(k) >=9
109             mintid2=[mintid2;redhorse.mintid(k)];
110         end
111     end
112     mintid1=min(mintid2);
113     maxtid1=max(mintid2);
114     medeltid=(mintid1+maxtid1)/2;
115
116     if isempty(medeltid)==1
117         medeltid=1;
118         mintid1=1;
119     end
120
121     for k=1:length(redhorse.resultat)
122         if redhorse.mintid(k) == 0
123             redhorse.mintid(k) = medeltid;

```

```

112         end
113     end
114
115     maxstartp = max(redhorse.startp);
116     maxpengar = max(redhorse.pengar);
117     maxvinstform = max(redhorse.vinstform);
118     maxtrivsel = max(redhorse.trivselfaktor);
119     distanser = unique(redhorse.distans);
120     distansdummy = zeros(length(redhorse.distans),1);
121
122     if length(distanser) > 1
123         mindistans = min(distanser);
124         index = find(redhorse.distans>mindistans);
125         distansdummy(index)=1;
126     end
127
128     minisuperhorse.loppid = [minisuperhorse.loppid;
129                             redhorse.loppid];
130     minisuperhorse.id = [minisuperhorse.id ; redhorse.id];
131     minisuperhorse.resultat = [minisuperhorse.resultat ;
132                               redhorse.resultat];
133     minisuperhorse.startnummer = [minisuperhorse.
134                                   startnummer ; redhorse.startnummer];
135     minisuperhorse.loppnummer = [minisuperhorse.loppnummer
136                                  ; redhorse.loppnummer];
137     minisuperhorse.distans = [minisuperhorse.distans ;
138                              redhorse.distans];
139     minisuperhorse.tid = [minisuperhorse.tid ; redhorse.tid
140                           ];
141     minisuperhorse.ntungbana = [minisuperhorse.ntungbana ;
142                                redhorse.ntungbana];
143     minisuperhorse.vinterbana = [minisuperhorse.vinterbana
144                                  ; redhorse.vinterbana];
145     minisuperhorse.tungbana = [minisuperhorse.tungbana ;
146                               redhorse.tungbana];
147     minisuperhorse.odds = [minisuperhorse.odds ; redhorse.
148                           odds];
149     minisuperhorse.kusk = [minisuperhorse.kusk redhorse.
150                           kusk];
151     minisuperhorse.trainer = [minisuperhorse.trainer
152                               redhorse.trainer];
153     minisuperhorse.autostart = [minisuperhorse.autostart ;
154                                 redhorse.autostart];
155     minisuperhorse.galopp = [minisuperhorse.galopp ;
156                              redhorse.galopp];
157     minisuperhorse.skorfram = [minisuperhorse.skorfram ;
158                                redhorse.skorfram];
159     minisuperhorse.skorbak = [minisuperhorse.skorbak ;
160                               redhorse.skorbak];

```

```

145 minisuperhorse.vinst = [minisuperhorse.vinst ; redhorse
    .vinst];
146 minisuperhorse.alder = [minisuperhorse.alder ; redhorse
    .alder];
147 minisuperhorse.datum = [minisuperhorse.datum ; redhorse
    .datum];
148 minisuperhorse.namn = [minisuperhorse. ; redhorse.namn
    ];
149 minisuperhorse.brakusk = [minisuperhorse.brakusk ;
    redhorse.brakusk];
150 minisuperhorse.ganskabrukusk = [minisuperhorse.
    ganskabrukusk ; redhorse.ganskabrukusk];
151 minisuperhorse.kuskrank = [minisuperhorse.kuskrank ;
    redhorse.kuskrank];
152 minisuperhorse.valack = [minisuperhorse.valack ;
    redhorse.valack];
153 minisuperhorse.hingst = [minisuperhorse.hingst ;
    redhorse.hingst];
154 minisuperhorse.sto = [minisuperhorse.sto ; redhorse.sto
    ];
155 minisuperhorse.mycketbrukusk = [minisuperhorse.
    mycketbrukusk ; redhorse.mycketbrukusk];
156 minisuperhorse.mktbrahorse = [minisuperhorse.
    mktbrahorse ; redhorse.mktbrahorse];
157 minisuperhorse.brahorse = [minisuperhorse.brahorse ;
    redhorse.brahorse];
158 minisuperhorse.ganskabrahorse = [minisuperhorse.
    ganskabrahorse ; redhorse.ganskabrahorse];
159 minisuperhorse.vinstprocent = [minisuperhorse.
    vinstprocent ; redhorse.vinstprocent];
160 minisuperhorse.startp = [minisuperhorse.startp ;
    redhorse.startp/maxstartp];
161 minisuperhorse.pengar = [minisuperhorse.pengar;
    redhorse.pengar/maxpengar];
162 minisuperhorse.vinstform = [minisuperhorse.vinstform;
    redhorse.vinstform/maxvinstform];
163 minisuperhorse.distansdummy = [minisuperhorse.
    distansdummy; distansdummy];
164 minisuperhorse.platsprocent = [minisuperhorse.
    platsprocent; redhorse.platsprocent];
165 minisuperhorse.trivselfaktor = [minisuperhorse.
    trivselfaktor; redhorse.trivselfaktor/maxtrivsel];
166 minisuperhorse.segerform = [minisuperhorse.segerform
    redhorse.segerform];
167 %minisuperhorse.cvek = [minisuperhorse.cvek; redhorse.
    cvek];
168 minisuperhorse.mintid = [minisuperhorse.mintid;
    redhorse.mintid/mintid1];

```

```

169 minisuperhorse.banavinsttrivsel=[minisuperhorse.
    banavinsttrivsel;redhorse.banavinsttrivsel];
170 minisuperhorse.banatopp3trivsel=[minisuperhorse.
    banatopp3trivsel;redhorse.banatopp3trivsel];
171 minisuperhorse.distansvinsttrivsel=[minisuperhorse.
    distansvinsttrivsel;redhorse.distansvinsttrivsel];
172 minisuperhorse.distanstopp3trivsel=[minisuperhorse.
    distanstopp3trivsel;redhorse.distanstopp3trivsel];
173 minisuperhorse.autovinsttrivsel=[minisuperhorse.
    autovinsttrivsel;redhorse.autovinsttrivsel];
174 minisuperhorse.autotopp3trivsel=[minisuperhorse.
    autotopp3trivsel;redhorse.autotopp3trivsel];
175 minisuperhorse.sommarvinsttrivsel=[minisuperhorse.
    sommarvinsttrivsel;redhorse.sommarvinsttrivsel] ;
176 minisuperhorse.sommartopp3trivsel=[minisuperhorse.
    sommartopp3trivsel;redhorse.sommartopp3trivsel] ;
177 end

```

Bilaga C

MATLAB Scripts - Models

```
1 %% Model 2: Find best covariates based on AIC
2 tic
3
4 load('minisuperhorse10') % Loading data
5 load('predictionhorse10')
6
7 Y = log(minisuperhorse.omvresultat');
8
9 minAICvec = []; minRMSvec = []; minAICvecPRED = [];
10 minRMSvecPRED = [];
11 covariates=[{'odds'} {'pengar'} {'alder' 'brakusk'} {'
12 mycketbrakusk'} {'ganskabrakusk'} {'mktbrahorse'}...
13 {'ganskabrahorse'} {'brahorse'} {'sto'} {'hingst'} {'
14 vinstprocent'} {'platsprocent'}...
15 {'startp'} {'vinstform'} {'segerform'} {'mintid'} {'
16 StartnummerRank'}...
17 {'distansvinsttrivsel'} {'distanstopp3trivsel'} {'
18 sommarvinsttrivsel'} {'sommartopp3trivsel'}...
19 {'autovinsttrivsel'} {'autotopp3trivsel'} {'distansdummy'}
20 {'autostart'}}];
21
22 covatiatematrixReg = [minisuperhorse.odds minisuperhorse.
23 pengar minisuperhorse.alder...
24 minisuperhorse.brakusk minisuperhorse.mycketbrakusk
25 minisuperhorse.ganskabrakusk...
26 minisuperhorse.mktbrahorse minisuperhorse.ganskabrahorse
27 minisuperhorse.brahorse...
28 minisuperhorse.sto minisuperhorse.hingst minisuperhorse.
29 vinstprocent...
30 minisuperhorse.platsprocent minisuperhorse.startp
31 minisuperhorse.vinstform...]
```

```

22 minisuperhorse.segerform' minisuperhorse.mintid
   minisuperhorse.StartnummerRank...
23 minisuperhorse.distansvinsttrivsel minisuperhorse.
   distanstopp3trivsel...
24 minisuperhorse.sommarvinsttrivsel minisuperhorse.
   sommartopp3trivsel...
25 minisuperhorse.autovinsttrivsel minisuperhorse.
   autotopp3trivsel...
26 minisuperhorse.distansdummy minisuperhorse.autostart];
27
28 Xreg = [];squared = [];sqroot = [];squaredcov = [];
   sqrootcov = [];multcov = [];alonecov = [];
29 antcov = length(covatiatematrixReg(1,:));ant = 1;vek = [];
30
31 for k = 1:antcov
32     alonecov = [alonecov; covariates(k) {'alone'}];
33     squared = [squared covatiatematrixReg(:,k).^2];
34     squaredcov = [squaredcov; covariates(k) {'squared'}];
35     sqroot = [sqroot covatiatematrixReg(:,k).^1/2];
36     sqrootcov = [sqrootcov; covariates(k) {'root'}];
37     for n = k+1:antcov
38         if sum(covatiatematrixReg(:,k).*covatiatematrixReg
   (:,n)) ~=0
39             Xreg=[Xreg covatiatematrixReg(:,k).*
   covatiatematrixReg(:,n)];
40             multcov = [multcov; covariates(k) covariates(n)
   ];
41         end
42     end
43 end
44
45 finalcovariates = [alonecov; squaredcov ;sqrootcov; multcov
   ];
46 covatiatematrixReg = [covatiatematrixReg sqroot squared
   Xreg];
47
48 covatiatematrixPred = [predictionhorse.odds predictionhorse
   .pengar predictionhorse.alder...
49 predictionhorse.brakusk predictionhorse.mycketbrakusk
   predictionhorse.ganskabrakusk...
50 predictionhorse.mktbrahorse predictionhorse.ganskabrahorse
   predictionhorse.brahorse...
51 predictionhorse.sto predictionhorse.hingst predictionhorse.
   vinstprocent...
52 predictionhorse.platsprocent predictionhorse.startp
   predictionhorse.vinstform...
53 predictionhorse.segerform predictionhorse.mintid
   predictionhorse.StartnummerRank...

```



```

54 predictionhorse.distansttrivsel predictionhorse.
    distanstopp3trivsel...
55 predictionhorse.sommarvinsttrivsel predictionhorse.
    sommartopp3trivsel...
56 predictionhorse.autovinsttrivsel predictionhorse.
    autotopp3trivsel...
57 predictionhorse.distanstdummy predictionhorse.autostart];
58
59 Xpred=[];squared = [];sqroot1 = [];
60
61 for t = 1:antcov
62     squared = [squared covatiatematrixPred(:,t).^2];
63     sqroot1 = [sqroot1 covatiatematrixPred(:,t).^1/2];
64     for n = t+1:antcov
65         if sum(covatiatematrixReg(:,t).*covatiatematrixReg
           (:,n)) ~=0
66             Xpred=[Xpred covatiatematrixPred(:,t).*
                covatiatematrixPred(:,n)];
67         end
68     end
69 end
70
71 covatiatematrixPred = [covatiatematrixPred sqroot1 squared
    Xpred];
72
73 AICvec=[];rmsvec=[];[rader kolonner] = size(
    covatiatematrixReg);XregFINAL = [];
74 XpredFINAL = [];used =[];okindex = 1:kolonner;n=1;
75
76 for k = 1:300
77     AICvec=[];
78     rmsvec=[];
79     for m =okindex
80         Xreg = XregFINAL;
81         antkol=m;
82         Xreg=[Xreg covatiatematrixReg(:,m)];
83         X=[ones(length(Y),1) Xreg];
84
85         b = regress(Y,X);
86         [AICvec(m) SSE rmsvec(m)] = modellanalys(b, log(
            minisuperhorse.omvresultat'), X);
87
88         if sum(find(b==0))>=1
89             AICvec(m)=10e20;
90         else
91             [AICvec(m) SSE rmsvec(m)] = modellanalys(b, log
                (minisuperhorse.omvresultat'), X);
92         end
93     n=n+1;

```

```

94 end
95 AICvec(used)=10e20;
96 a=find(AICvec == min(AICvec));
97 used =[used;a(1)];
98 XregFINAL = [XregFINAL covatiatematrixReg(:,a(1))];
99 XpredFINAL = [XpredFINAL covatiatematrixPred(:,a(1))];
100 okindex(find(okindex==a(1)))=[];
101 minAICvec = [minAICvec; min(AICvec)];
102 minRMSvec = [minRMSvec; rmsvec(a(1))];
103 X=[ones(length(Y),1) XregFINAL];
104 b = regress(Y,X);
105 [AIC SSE rms] = modellanalys(b, Y, X);
106 minAICvecPRED = [minAICvecPRED; AIC];
107 minRMSvecPRED = [minRMSvecPRED; rms];
108 [minAICvec minAICvecPRED]
109 end
110 toc

```

```

1 %% Modell 2
2
3 load minisuperhorse10INKLloppid % Loading data
4 load predictionhorse10
5
6 covariates=[{'odds'} {'pengar'} {'alder' 'brakusk'} {'
    mycketbrakusk'} {'ganskabrakusk'} {'mktbrahorse'}...
7 {'ganskabrahorse'} {'brahorse'} {'sto'} {'hingst'} {'
    vinstprocent'} {'platsprocent'}...
8 {'startp'} {'vinstform'} {'segerform'} {'mintid'} {'
    StartnummerRank'}...
9 {'distansvinsttrivsel'} {'distanstopp3trivsel'} {'
    sommarvinsttrivsel'} {'sommartopp3trivsel'}...
10 {'autovinsttrivsel'} {'autotopp3trivsel'} {'distansdummy'}
    {'autostart'}}];
11
12 covatiatematrixReg = [minisuperhorse.odds minisuperhorse.
    pengar minisuperhorse.alder...
13 minisuperhorse.brakusk minisuperhorse.mycketbrakusk
    minisuperhorse.ganskabrakusk...
14 minisuperhorse.mktbrahorse minisuperhorse.ganskabrahorse
    minisuperhorse.brahorse...
15 minisuperhorse.sto minisuperhorse.hingst minisuperhorse.
    vinstprocent...
16 minisuperhorse.platsprocent minisuperhorse.startp
    minisuperhorse.vinstform...
17 minisuperhorse.segerform' minisuperhorse.mintid
    minisuperhorse.StartnummerRank...
18 minisuperhorse.distansvinsttrivsel minisuperhorse.
    distanstopp3trivsel...

```

```

19 minisuperhorse.sommarvinsttrivsel minisuperhorse.
   sommartopp3trivsel...
20 minisuperhorse.autovinsttrivsel minisuperhorse.
   autotopp3trivsel...
21 minisuperhorse.distansdummy minisuperhorse.autostart];
22
23 covariatematrixPred = [predictionhorse.odds predictionhorse
   .pengar predictionhorse.alder...
24 predictionhorse.brakusk predictionhorse.mycketbrakusk
   predictionhorse.ganskabrakusk...
25 predictionhorse.mktbrahorse predictionhorse.ganskabrahorse
   predictionhorse.brahorse...
26 predictionhorse.sto predictionhorse.hingst predictionhorse.
   vinstprocent...
27 predictionhorse.platsprocent predictionhorse.startp
   predictionhorse.vinstform...
28 predictionhorse.segerform predictionhorse.mintid
   predictionhorse.StartnummerRank...
29 predictionhorse.distansvinsttrivsel predictionhorse.
   distanstopp3trivsel...
30 predictionhorse.sommarvinsttrivsel predictionhorse.
   sommartopp3trivsel...
31 predictionhorse.autovinsttrivsel predictionhorse.
   autotopp3trivsel...
32 predictionhorse.distansdummy predictionhorse.autostart];
33
34 squared = [];sqroot = [];squaredcov = [];sqrootcov = [];
   multcov = [];alonecov = [];
35 antcov = length(covariatematrixReg(1,:));RMSdata=[];ant =
   1;vek = [];Xreg=[];
36
37 for k = 1:antcov
38     alonecov = [alonecov; covariates(k) {'alone'}];
39     squared = [squared covariatematrixReg(:,k).^2];
40     squaredcov = [squaredcov; covariates(k) {'squared'}];
41     sqroot = [sqroot covariatematrixReg(:,k).^1/2];
42     sqrootcov = [sqrootcov; covariates(k) {'root'}];
43     for n = k+1:antcov
44         if sum(covariatematrixReg(:,k).*covariatematrixReg
           (:,n)) ~=0
45             Xreg=[Xreg covariatematrixReg(:,k).*
               covariatematrixReg(:,n)];
46             multcov = [multcov; covariates(k) covariates(n)
               ];
47         end
48     end
49 end
50

```

```

51 finalcovariates = [alonecov; squaredcov ;sqrootcov; multcov
    ];
52 covariatematrixReg = [covariatematrixReg sqroot squared
    Xreg];
53
54 Xpred=[];squared = [];sqroot1 = [];
55
56 for t = 1:antcov
57     squared = [squared covariatematrixPred(:,t).^2];
58     sqroot1 = [sqroot1 covariatematrixPred(:,t).^1/2];
59     for n = t+1:antcov
60         if sum(covariatematrixReg(:,t).*covariatematrixReg
           (:,n)) ~=0
61             Xpred=[Xpred covariatematrixPred(:,t).*
                covariatematrixPred(:,n)];
62         end
63     end
64 end
65 covariatematrixPred = [covariatematrixPred sqroot1 squared
    Xpred];
66
67 % Dependent variable
68 Y = log(minisuperhorse.omvresultat');
69 Ylogistisk = zeros(length(Y),1);
70 for i = 1:length(Y)
71     if minisuperhorse.omvresultat(i)==1
72         Ylogistisk(i)=1;
73     end
74 end
75
76 load used
77 used=used(1:35); % Covariates
78 Xreg=covariatematrixReg(:,used);
79
80 % Regression
81 beta = Xreg\Y;
82
83 % Predict
84 Xpred=covariatematrixPred(:,used);
85 prediktion = exp(Xpred*beta);
86 pred2 = prediktion;
87 [dagskassa antalhorses jmntmatris SOS SOSo] = analys(
    predictionhorse,prediktion);
88 plot(1:length(dagskassa),dagskassa,1:length(dagskassa),
    dagskassa,'r.')
89
90 % Calculate RMSE for the model
91 unika=unique(minisuperhorse.loppid);
92 p=[];

```

```

93 Xreg=covariatematrixReg(:,used);
94 beta = Xreg\Y;
95 prediktion=exp(Xreg*beta);
96
97 for i=unika'
98     plats=find(minisuperhorse.loppid==i);
99     predtemp=prediktion(plats);
100     ptemp = exp2prob(predtemp');
101     p=[p;ptemp(:,1)];
102 end
103
104 SSE=sum((Ylogistisk-p).^2);
105 RMSE=sqrt(SSE/length(Ylogistisk));
106
107 SSE=sum((Ylogistisk-prediktiondataNormerad).^2);
108 RMSElogistisk=sqrt(SSE/length(Ylogistisk));
109
110 %% Predict seven V75 days
111 datvektor=[130413 130406 130223 130105 121201 121020 121013
112     ];
113 idvektor=[529859 530004 529394 529398 521831 521823 522118
114     ];
115 SOS=0;
116 SOSo=0;
117
118 load Predictionhorse %Data
119
120 dagskassa=1000;
121 for i=1:7
122     predictionhorse=Predictionhorse(i);
123     covariatematrixPred = [predictionhorse.odds
124         predictionhorse.pengar predictionhorse.alder...
125         predictionhorse.brakusk predictionhorse.mycketbrakusk
126         predictionhorse.ganskabrakusk...
127         predictionhorse.mktbrahorse predictionhorse.
128         ganskabrahorse predictionhorse.brahorse...
129         predictionhorse.sto predictionhorse.hingst
130         predictionhorse.vinstprocent...
131         predictionhorse.platsprocent predictionhorse.startp
132         predictionhorse.vinstform...
133         predictionhorse.segerform predictionhorse.mintid
134         predictionhorse.StartnummerRank...
135         predictionhorse.distansvinsttrivsel predictionhorse.
136         distanstopp3trivsel...
137         predictionhorse.sommarvinsttrivsel predictionhorse.
138         sommartopp3trivsel...
139         predictionhorse.autovinsttrivsel predictionhorse.
140         autotopp3trivsel...

```

```

130 predictionhorse.distansdummy predictionhorse.autostart
    ];
131 squared = [];sqroot = [];squaredcov = [];sqrootcov =
    [];multcov = [];alonecov = [];
132 antcov = length(covariatematrixPred(1,:));RMSdata=[];
    ant = 1;vek = [];Xpred=[];
133
134     for k = 1:antcov
135         squared = [squared covariatematrixPred(:,k)
136             .^2];
137         sqroot = [sqroot covariatematrixPred(:,k)
138             .^1/2];
139         for n = k+1:antcov
140             if sum(covariatematrixReg(:,k).*
141                 covariatematrixReg(:,n)) ~=0
142                 Xpred=[Xpred covariatematrixPred(:,k).*
143                     covariatematrixPred(:,n)];
144             end
145         end
146     end
147
148     finalcovariates = [alonecov; squaredcov ;sqrootcov;
149         multcov;];
150     covariatematrixPred = [covariatematrixPred sqroot
151         squared Xpred];
152     Xpred=covariatematrixPred(:,used);
153     prediktion = exp(Xpred*beta);
154     [dagskassa SOSp SOSo] = analysmedopt(dagskassa,
155         predictionhorse,prediktion);
156     dagskassa
157 end

```

```

1 function [dagskassa antalhorses jmntmatris SOS SOSo]=
    analys(predictionhorse, prediktion)
2 SOS = 0;
3 SOSo = 0;
4 SOS3 = 0;
5 SOSo3 = 0;
6 SOS1 = 0;
7 SOSo1 = 0;
8
9 % Real result
10 resultat = predictionhorse.omvresultat';
11 odds=predictionhorse.odds;
12
13 unikloppid=unique(predictionhorse.loppid);
14 idmatris = zeros(15,7);
15 idrankodds = zeros(15,7);
16 idrankpred = zeros(15,7);

```

```

17 predrankvinnare = [];
18 oddsrankvinnare = [];
19 antalhorses = [];
20 kassa=1000;
21 dagskassa=[kassa];
22 sannolikhet=[];
23
24 for j=1:length(unkiloppid)
25     plats=find(predictionhorse.loppid==unkiloppid(j));
26     rank=tiedrank(prediktion(plats));
27     prediktiontemp = prediktion(plats);
28     tiedrankodds=tiedrank(odds(plats));
29     odds1=odds(plats)/10;
30     prediktion1=prediktion(plats);
31     resultat1=resultat(plats);
32     prob = exp2prob(prediktiontemp');
33     prob = prob(:,1);
34     odds2 = 1./odds1;
35     probodds = odds2./(sum(odds2));
36     sannolikhet=[sannolikhet; prob];
37
38     m=[resultat1' tiedrankodds rank odds1 prediktiontemp/
39       sum(prediktiontemp)*sum(1:length(rank)) prob(:,1)
40       probodds prob(:,1)-odds2];
41     [B I]=sort(m(:,1));
42     m2=m(I,:);
43     disp(m2)
44
45     asd=find((prob-odds2)>0);
46     betprob=max(prob(asd));
47     if isempty(betprob)==0
48         k=find(betprob==prob);
49         oddsbet=odds1(k);
50         if betprob > 0.2
51             summa = 200;
52         else
53             summa=100;
54         end
55         if betprob>0.1
56             placering=resultat1(k);
57             if placering==1
58                 kassa=kassa+summa*(odds1(k)-1);
59             else
60                 kassa=kassa-summa;
61             end
62         end
63     end
64     dagskassa=[dagskassa; kassa];
65 end

```

```

1 function [antalhorses jmntmatrix SOS SOSo]= analysmedopt(
   predictionhorse, prediktion)
2 % Real result
3 resultat = predictionhorse.omvresultat';
4 odds=predictionhorse.odds;
5
6 unkiloppid=unique(predictionhorse.loppid);
7
8 idmatrix = zeros(15,7);idrankodds = zeros(15,7);idrankpred
   = zeros(15,7);predrankvinnare = [];
9 oddsrankvinnare = [];antalhorses = [];jmnt = [];jmnt2 = [];
   jmnt3 = [];jmnt4 = [];jmnt5 = [];
10 jmnt6 = [];jmnt7 = [];jmntmatrix = [];probmatrix=[];
   probunsorted=[];
11 proboddsmatrix=[];oddsmatrix=[];sannolikhet=[];
12
13 for j=1:length(unkiloppid)
14     plats=find(predictionhorse.loppid==unkiloppid(j));
15     rank=tiedrank(prediktion(plats));
16     prediktiontemp = prediktion(plats);
17     tiedrankodds=tiedrank(odds(plats));
18     odds1=odds(plats)/10;
19     prediktion1=prediktion(plats);
20     resultat1=resultat(plats);
21     prob = exp2prob(prediktiontemp');
22     prob = prob(:,1);
23     odds2 = 1./odds1;
24     probodds = odds2./(sum(odds2));
25     sannolikhet=[sannolikhet; prob];
26
27     probvec = sort(prob(:,1),'descend');
28     jmntvec = [];
29     for k = 1:length(probvec)
30         jmntvec = [jmntvec sum(probvec(1:k))];
31     end
32
33     jmntvec=[jmntvec zeros(1,15-length(jmntvec))];
34     jmntmatrix = [jmntmatrix; jmntvec];
35     antalhorses = [antalhorses length(rank)];
36     jmntvec=[jmntvec zeros(1,15-length(jmntvec))];
37     jmntmatrix = [jmntmatrix; jmntvec];
38     antalhorses = [antalhorses length(rank)];
39     probvec15 = [probvec zeros(1,15-length(probvec))];
40     probmatrix = [probmatrix; probvec15];
41     probunsortedtemp = [prob(:,1)' zeros(1,15-length(prob
   (:,1)))];
42     probunsorted = [probusorted; probunsortedtemp];

```



```

43 proboddstemp = [probodds' zeros(1,15-length(probodds
    (:,1)))];
44 proboddsmatrix = [proboddsmatrix; proboddstemp ];
45 oddstemp = [odds1' zeros(1,15-length(odds1))];
46 oddsmatrix = [oddsmatrix; oddstemp];
47
48 idmatrix(1:length(predictionhorse.id(plats(j):plats(j
    +1)-1)),j) = predictionhorse.id(plats(j):plats(j+1)
    -1);
49 rankpred=nontiedrank(prediktion(plats(j):plats(j+1)-1))
    ;
50 rankodds=nontiedrank(odds(plats(j):plats(j+1)-1));
51 jmnt2 = [jmnt2; prediktiontemp(find(rankpred==2)) -
    prediktiontemp(find(rankpred==1))];
52 jmnt3 = [jmnt3; prediktiontemp(find(rankpred==3)) -
    prediktiontemp(find(rankpred==1))];
53 jmnt4 = [jmnt4; prediktiontemp(find(rankpred==4)) -
    prediktiontemp(find(rankpred==1))];
54 jmnt5 = [jmnt5; prediktiontemp(find(rankpred==5)) -
    prediktiontemp(find(rankpred==1))];
55 jmnt6 = [jmnt6; prediktiontemp(find(rankpred==6)) -
    prediktiontemp(find(rankpred==1))];
56 jmnt7 = [jmnt7; prediktiontemp(find(rankpred==7)) -
    prediktiontemp(find(rankpred==1))];
57 predrankvinnare = [predrankvinnare; rank(1)];
58 oddsrankvinnare = [oddsrankvinnare; tiedrankodds(1)];
59 end
60 optimeringspelsystem

```

```

1 %% Optimizing V75
2 load vec_400-500 % Loading different combinations
3
4 vec = [];
5 antalhorses=[10 12 13 14 15 12 12];
6 for a = 1:min(antalhorses(1), 9)
7     for b = 1:min(antalhorses(2), 9)
8         for c = 1:min(antalhorses(3), 9)
9             for d = 1:min(antalhorses(4), 9)
10                 for e = 1:min(antalhorses(5), 9)
11                     for f = 1:min(antalhorses(6), 9)
12                         for g = 1:min(antalhorses(7), 9)
13                             if a*b*c*d*e*f*g >=400*2 && a*b
                                *c*d*e*f*g <= 500*2
14                                 vec = [vec; a b c d e f g];
15                             end
16                         end
17                     end
18                 end
19             end

```

```

20         end
21     end
22 end
23
24 vinstvec = [];vinstvec04 = [];vinstvec05 = [];vinstvec06 =
    [];vinstvec07 = [];
25
26 for j=1:length(vec)
27     vinstvec= [vinstvec; jmntmatrix(1,vec(j,1))*jmntmatrix
        (2,vec(j,2))*...
28     jmntmatrix(3,vec(j,3))*jmntmatrix(4,vec(j,4))*
        jmntmatrix(5,vec(j,5))*...
29     jmntmatrix(6,vec(j,6))*jmntmatrix(7,vec(j,7))];
30     A = [jmntmatrix(1,vec(j,1)) jmntmatrix(2,vec(j,2))
        jmntmatrix(3,vec(j,3)) jmntmatrix(4,vec(j,4))...
31     jmntmatrix(5,vec(j,5)) jmntmatrix(6,vec(j,6))
        jmntmatrix(7,vec(j,7))];
32     index = find(A>=0.4);
33
34     numberOfElements = length(index);
35     if numberOfElements > 6
36         vinstvec04 = [vinstvec04; jmntmatrix(1,vec(j,1))*
            jmntmatrix(2,vec(j,2))*...
37         jmntmatrix(3,vec(j,3))*jmntmatrix(4,vec(j,4))*
            jmntmatrix(5,vec(j,5))*...
38         jmntmatrix(6,vec(j,6))*jmntmatrix(7,vec(j,7))];
39     else
40         vinstvec04 = [vinstvec04; 0];
41     end
42
43     index = find(A>=0.5);
44     numberOfElements = length(index);
45     if numberOfElements > 6
46         vinstvec05 = [vinstvec05; jmntmatrix(1,vec(j,1))*
            jmntmatrix(2,vec(j,2))*...
47         jmntmatrix(3,vec(j,3))*jmntmatrix(4,vec(j,4))*
            jmntmatrix(5,vec(j,5))*...
48         jmntmatrix(6,vec(j,6))*jmntmatrix(7,vec(j,7))];
49     else
50         vinstvec05 = [vinstvec05; 0];
51     end
52
53     index = find(A>=0.6);
54     numberOfElements = length(index);
55     if numberOfElements > 6
56         vinstvec06 = [vinstvec06; jmntmatrix(1,vec(j,1))*
            jmntmatrix(2,vec(j,2))*...
57         jmntmatrix(3,vec(j,3))*jmntmatrix(4,vec(j,4))*
            jmntmatrix(5,vec(j,5))*...

```

```

58         jmntmatrix(6,vec(j,6))*jmntmatrix(7,vec(j,7))];
59     else
60         vinstvec06 = [vinstvec06; 0];
61     end
62
63     index = find(A>=0.7);
64     numberOfElements = length(index);
65     if numberOfElements > 6
66         vinstvec07 = [vinstvec07; jmntmatrix(1,vec(j,1))*
67             jmntmatrix(2,vec(j,2))*...
68             jmntmatrix(3,vec(j,3))*jmntmatrix(4,vec(j,4))*
69             jmntmatrix(5,vec(j,5))*...
70             jmntmatrix(6,vec(j,6))*jmntmatrix(7,vec(j,7))];
71     else
72         vinstvec07 = [vinstvec07; 0];
73     end
74
75     % Win often
76     slh= max(vinstvec);slh04 = max(vinstvec04);slh05 = max(
77         vinstvec05);slh06 = max(vinstvec06);
78     slh07 = max(vinstvec07);a=find(vinstvec==slh);spel=vec(a,:)
79     ;
80     if slh04 > 0
81         a=find(vinstvec04==slh04);
82         spel04=vec(a,:);
83     else
84         spel04 =[];
85     end
86
87     if slh05 > 0
88         a=find(vinstvec05==slh05);
89         spel05=vec(a,:);
90     else
91         spel05 =[];
92     end
93
94     if slh06 > 0
95         a=find(vinstvec06==slh06);
96         spel06=vec(a,:);
97     else
98         spel06 =[];
99     end
100
101     if slh07 > 0
102         a=find(vinstvec07==slh07);
103         spel07=vec(a,:);
104     else
105         spel07 =[];

```

103 end

```
1 %% Model 3: Decide wich covariates to include based on ROC
2
3 load('minisuperhorse10') %Load data set
4 load('predictionhorse10')
5
6 Y = zeros(length(minisuperhorse.resultat),1);
7
8 for j = 1:length(Y)
9     if minisuperhorse.resultat(j)==1
10         Y(j)=1;
11     end
12 end
13
14 AUCvecPRED = []; maxAUCvec = [];
15
16 covariates=[{'odds'} {'pengar'} {'alder'} {'brakusk'} {'
17     mycketbrakusk'} {'ganskabrakusk'} {'mktbrahorse'}...
18 {'ganskabrahorse'} {'brahorse'} {'sto'} {'hingst'} {'
19     vinstprocent'} {'platsprocent'}...
20 {'startp'} {'vinstform'} {'segerform'} {'mintid'} {'
21     StartnummerRank'}...
22 {'distansvinsttrivsel'} {'distanstopp3trivsel'} {'
23     sommarvinsttrivsel'} {'sommartopp3trivsel'}...
24 {'autovinsttrivsel'} {'autotopp3trivsel'} {'distansdummy'}
25     {'autostart'}}];
26
27 covatiatematrixReg = [minisuperhorse.odds minisuperhorse.
28     pengar minisuperhorse.alder...
29     minisuperhorse.brakusk minisuperhorse.mycketbrakusk
30     minisuperhorse.ganskabrakusk...
31     minisuperhorse.mktbrahorse minisuperhorse.ganskabrahorse
32     minisuperhorse.brahorse...
33     minisuperhorse.sto minisuperhorse.hingst minisuperhorse.
34     vinstprocent...
35     minisuperhorse.platsprocent minisuperhorse.startp
36     minisuperhorse.vinstform...
37     minisuperhorse.segerform' minisuperhorse.mintid
38     minisuperhorse.StartnummerRank...
39     minisuperhorse.distansvinsttrivsel minisuperhorse.
40     distanstopp3trivsel...
41     minisuperhorse.sommarvinsttrivsel minisuperhorse.
42     sommartopp3trivsel...
43     minisuperhorse.autovinsttrivsel minisuperhorse.
44     autotopp3trivsel...
45     minisuperhorse.distansdummy minisuperhorse.autostart];
```

```

33 RMSpred = []; RMSvecdata = []; RMSvecpred = []; Xreg = [];
    squared = [];
34 sqroot = []; squaredcov = []; sqrootcov = []; multcov = [];
    alonecov = [];
35 antcov = length(covatiatematrixReg(1,:)); ant = 1; vek = [];
36
37 for k = 1:antcov
38     alonecov = [alonecov; covariates(k) {'alone'}];
39     squared = [squared covatiatematrixReg(:,k).^2];
40     squaredcov = [squaredcov; covariates(k) {'squared'}];
41     sqroot = [sqroot covatiatematrixReg(:,k).^1/2];
42     sqrootcov = [sqrootcov; covariates(k) {'root'}];
43
44     for n = k+1:antcov
45         if sum(covatiatematrixReg(:,k).*covatiatematrixReg
46             (:,n)) ~=0
47             Xreg=[Xreg covatiatematrixReg(:,k).*
48                 covatiatematrixReg(:,n)];
49             multcov = [multcov; covariates(k) covariates(n)
50                 ];
51         end
52     end
53 end
54
55 finalcovariates = [alonecov; squaredcov ;sqrootcov; multcov
56     ];
57 covatiatematrixReg = [covatiatematrixReg sqroot squared];
58
59 covatiatematrixPred = [predictionhorse.odds predictionhorse
60     .pengar predictionhorse.alder...
61     predictionhorse.brakusk predictionhorse.mycketbrakusk
62     predictionhorse.ganskabrakusk...
63     predictionhorse.mktbrahorse predictionhorse.ganskabrahorse
64     predictionhorse.brahorse...
65     predictionhorse.sto predictionhorse.hingst predictionhorse.
66     vinstprocent...
67     predictionhorse.platsprocent predictionhorse.startp
68     predictionhorse.vinstform...
69     predictionhorse.segerform predictionhorse.mintid
70     predictionhorse.StartnummerRank...
71     predictionhorse.distansvinsttrivsel predictionhorse.
72     distanstopp3trivsel...
73     predictionhorse.sommarvinsttrivsel predictionhorse.
74     sommartopp3trivsel...
75     predictionhorse.autovinsttrivsel predictionhorse.
76     autotopp3trivsel...
77     predictionhorse.distansdummy predictionhorse.autostart];
78
79 Xpred=[]; squared = []; sqroot1 = [];

```

```

67
68 for t = 1:antcov
69     squared = [squared covatiatematrixPred(:,t).^2];
70     sqroot1 = [sqroot1 covatiatematrixPred(:,t).^1/2];
71
72     for n = t+1:antcov
73         if sum(covatiatematrixReg(:,t).*covatiatematrixReg
74             (:,n)) ~=0
75             Xpred=[Xpred covatiatematrixPred(:,t).*
76                 covatiatematrixPred(:,n)];
77         end
78     end
79 end
80
81 covatiatematrixPred = [covatiatematrixPred sqroot1 squared
82     ];
83
84 AUCvec=[];used =[];
85 [rader kolonner] = size(covatiatematrixReg);
86 XregFINAL = covatiatematrixReg(:,used);
87 XpredFINAL = [];RMSdata=[];
88 okindex = 1:kolonner;
89
90 for k = 1:300
91     AUCvec=[];
92     for m =okindex
93         Xreg = XregFINAL;
94         antkol=m;
95         Xreg=[Xreg covatiatematrixReg(:,m)];
96         b = glmfit(Xreg,Y,'binomial','link','logit');
97         Xpred=[covatiatematrixPred(:,used)
98             covatiatematrixPred(:,m)];
99         prediktion = exp([ones(length(predictionhorse.odds)
100             ,1) Xpred]*b)./(1+exp([ones(length(
101             predictionhorse.odds),1) Xpred]*b)));
102
103         a1 = predictionhorse.resultat;
104         b1 = prediktion;
105
106         Y1= zeros(length(a1),1);
107         for o=1:length(a1)
108             if a1(o)==1
109                 Y1(o)=1;
110             end
111         end
112
113         p = glmval(b,Xreg,'logit');
114         if sum(find(b==0))>=1
115             AUC=0;
116         end
117     end
118 end

```

```

110         else
111             [x,y,T,AUC ,OPTROCPT ,SUBY ,SUBYNAMES]=perfcurve(Y
                ,p,1); % Calc AUC
112         end
113
114         AUCvec(m)=AUC;
115     end
116
117     a=find(AUCvec ==max(AUCvec));
118     used =[used;a(1)];
119     XregFINAL = [XregFINAL covatiatematrixReg(:,a(1))];
120     b = glmfit(XregFINAL,Y,'binomial','link','logit');
121     Xpred=covatiatematrixPred(:,used);
122     prediktion = exp([ones(length(predictionhorse.odds),1)
        Xpred]*b)./(1+exp([ones(length(predictionhorse.odds)
        ,1) Xpred]*b));
123     [x,y,T,AUC ,OPTROCPT ,SUBY ,SUBYNAMES]=perfcurve(Y1 ,
        prediktion,1);
124     AUCvecPRED=[ AUCvecPRED; AUC];
125     SSEpred=sum((Y1-prediktion).^2);
126     n=length(Y1);
127     RMSvecpred=[RMSvecpred; sqrt(SSEpred/n)];
128     p = glmval(b,XregFINAL,'logit');
129     SSEdata=sum((Y-p).^2);
130     n=length(Y);
131     RMSdata = [RMSdata; sqrt(SSEdata/n)];
132     okindex(find(okindex==a(1)))=[];
133     maxAUCvec = [maxAUCvec; max(AUCvec)]
134     [RMSdata(end) RMSvecpred(end) used(end)]
135 end
136 toc

```

```

1 %% Model 3
2 close all, clc, clear all
3
4 % Data
5 load predictionhorse10
6 load minisuperhorse10
7 load used
8
9 % Logistic result
10 minisuperhorse.logisticresultat=zeros(length(minisuperhorse
    .resultat),1);
11 plats=find(minisuperhorse.resultat==1);
12 minisuperhorse.logisticresultat(plats)=1;
13
14 % Dependent variable
15 Y = minisuperhorse.logisticresultat;
16

```

```

17 % Covariates
18 covatiatematrixReg = [minisuperhorse.odds minisuperhorse.
    pengar minisuperhorse.alder...
19 minisuperhorse.brakusk minisuperhorse.mycketbrakusk
    minisuperhorse.ganskabrakusk...
20 minisuperhorse.mktbrahorse minisuperhorse.ganskabrahorse
    minisuperhorse.brahorse...
21 minisuperhorse.sto minisuperhorse.hingst minisuperhorse.
    vinstprocent...
22 minisuperhorse.platsprocent minisuperhorse.startp
    minisuperhorse.vinstform...
23 minisuperhorse.segerform' minisuperhorse.mintid
    minisuperhorse.StartnummerRank...
24 minisuperhorse.distansvinsttrivsel minisuperhorse.
    distanstopp3trivsel...
25 minisuperhorse.sommarvinsttrivsel minisuperhorse.
    sommartopp3trivsel...
26 minisuperhorse.autovinsttrivsel minisuperhorse.
    autotopp3trivsel...
27 minisuperhorse.distansdummy minisuperhorse.autostart];
28
29 covariates=[{'odds'} {'pengar'} {'alder'} {'brakusk'} {'
    mycketbrakusk'} {'ganskabrakusk'} {'mktbrahorse'}...
30 {'ganskabrahorse'} {'brahorse'} {'sto'} {'hingst'} {'
    vinstprocent'} {'platsprocent'}...
31 {'startp'} {'vinstform'} {'segerform'} {'mintid'} {'
    StartnummerRank'}...
32 {'distansvinsttrivsel'} {'distanstopp3trivsel'} {'
    sommarvinsttrivsel'} {'sommartopp3trivsel'}...
33 {'autovinsttrivsel'} {'autotopp3trivsel'} {'distansdummy'}
    {'autostart'}}];
34
35 % X Regression
36 Xreg = [];squared = [];sqroot = [];
37 squaredcov = [];sqrootcov = [];multcov = [];alonecov = [];
38 antcov = length(covatiatematrixReg(1,:));ant = 1;vek = [];
39
40 for k = 1:antcov
41     alonecov = [alonecov; covariates(k) {'alone'}];
42     squared = [squared covatiatematrixReg(:,k).^2];
43     squaredcov = [squaredcov; covariates(k) {'squared'}];
44     sqroot = [sqroot covatiatematrixReg(:,k).^1/2];
45     sqrootcov = [sqrootcov; covariates(k) {'root'}];
46     for n = k+1:antcov
47         if sum(covatiatematrixReg(:,k).*covatiatematrixReg
            (:,n)) ~=0
48             Xreg=[Xreg covatiatematrixReg(:,k).*
                covatiatematrixReg(:,n)];

```



```

49         multcov = [multcov; covariates(k) covariates(n)
50                     ];
51     end
52 end
53
54 finalcovariates = [alonecov; squaredcov ;sqrootcov; multcov
55                    ];
56 covatiatematrixReg = [covatiatematrixReg sqroot squared
57                      Xreg];
58 X=covatiatematrixReg(:,used);
59
60 % Regression
61 b = glmfit(X,Y,'binomial','link','logit');
62
63 % X Prediction
64 covatiatematrixPred = [predictionhorse.odds predictionhorse
65                        .pengar predictionhorse.alder...
66                        predictionhorse.brakusk predictionhorse.mycketbrakusk
67                        predictionhorse.ganskabrakusk...
68                        predictionhorse.mktbrahorse predictionhorse.ganskabrahorse
69                        predictionhorse.brahorse...
70                        predictionhorse.sto predictionhorse.hingst predictionhorse.
71                        vinstprocent...
72                        predictionhorse.platsprocent predictionhorse.startp
73                        predictionhorse.vinstform...
74                        predictionhorse.segerform predictionhorse.mintid
75                        predictionhorse.StartnummerRank...
76                        predictionhorse.distansvinsttrivsel predictionhorse.
77                        distanstopp3trivsel...
78                        predictionhorse.sommarvinsttrivsel predictionhorse.
79                        sommartopp3trivsel...
80                        predictionhorse.autovinsttrivsel predictionhorse.
81                        autotopp3trivsel...
82                        predictionhorse.distansdummy predictionhorse.autostart];
83
84 Xpred=[];squared = [];sqroot1 = [];
85 for t = 1:antcov
86     squared = [squared covatiatematrixPred(:,t).^2];
87     sqroot1 = [sqroot1 covatiatematrixPred(:,t).^1/2];
88     for n = t+1:antcov
89         if sum(covatiatematrixReg(:,t).*covatiatematrixReg
90                (:,n)) ~=0
91             Xpred=[Xpred covatiatematrixPred(:,t).*
92                    covatiatematrixPred(:,n)];
93         end
94     end
95 end

```

```

84 covatiatematrixPred = [covatiatematrixPred sqrt(1 + squared
    Xpred)];
85 X=[ones(length(predictionhorse.odds),1) covatiatematrixPred
    (:,used)];
86
87 % Prediction
88 prediktion = exp(X*b)./(1+exp(X*b));
89
90 % Analys
91 kassa=1000;
92 dagskassa = analysLogistisk(predictionhorse,prediktion,
    kassa);
93
94 % Cash plot
95 m=1000;
96 k=(dagskassa(end)-m)/length(dagskassa);
97 kx=1:length(dagskassa);
98 y=k*kx+m;
99 plot(kx,dagskassa,kx,dagskassa,'bo',kx,y,'r')
100 title('Betting strategy 3')
101 xlabel('Number of races ')
102 ylabel('SEK')

```

```

1 function dagskassa = analysLogistisk(predictionhorse,
    prediktion, kassa)
2 dagskassa=[];
3
4 % Real result
5 resultat = predictionhorse.omvresultat';
6 odds=predictionhorse.odds;
7
8 % Find races
9 unkiloppid=unique(predictionhorse.loppid);
10
11 % Loop for each race
12 for j=1:length(unkiloppid)
13     plats=find(predictionhorse.loppid==unkiloppid(j));
14     rank=tiedrank(-prediktion(plats));
15     prediktiontemp = prediktion(plats);
16     tiedrankodds=tiedrank(odds(plats));
17     odds1=odds(plats)/10;
18     resultat1=resultat(plats);
19
20 % Predicted prob
21 prob = prediktiontemp/(sum(prediktiontemp));
22
23 % Prob odds
24 odds2 = 1./odds1;
25 probodds = odds2./(sum(odds2));

```

```

26
27 % Disp result
28 m=[resultat1' tiedrankodds odds1 rank prediktiontemp
    prob(:,1) probodds prob(:,1)-odds2];
29 [B I]=sort(m(:,1));
30 m2=m(I,:);
31 disp(m2)
32
33 % Cash
34 asd=find((prob-odds2)>0);
35 betprob=max(prob(asd))
36 if isempty(betprob)==0
37 k=find(betprob==prob);
38 oddsbet=odds1(k)
39
40 if betprob > 0.2
41     summa=200;
42 else
43     summa=100;
44 end
45
46 if betprob>0.1
47     placering=resultat1(k)
48     if placering==1
49         kassa=kassa+summa*(odds1(k)-1);
50     else
51         kassa=kassa - summa;
52     end
53 end
54 end
55 dagskassa=[dagskassa; kassa];
56 end

```