Master Thesis Business Analytics

Predicting football results with an evolutionary ensemble classifier

Author:
Vincent Hoekstra

Supervisors:
Pieter Bison
Guszti Eiben

VU University Amsterdam
Faculty of Sciences
De Boelelaan 1081a
1081 HV Amsterdam

Sentient Information Systems
Singel 160
1015 AH Amsterdam

October, 2012

Acknowledgments

First and foremost, I would like to thank Marten den Uyl and Pieter Bison for the opportunity to do this research at Sentient and for their support during this research. I would also like to thank all the colleagues at SMR for the pleasant atmosphere at the company. I am particularly grateful for the assistance given by Bas Weitjens, for the help with DataDetective, Matthijs Mulder, for helping me with retrieving useful football information, and Jan Lasek, who helped me with the benchmarks for the research.

I also want to give my special thanks to my supervisor from the VU University, Guszti Eiben, who gave invaluable tips throughout the course of the internship, that always turned out to be exactly right and very important. I would also like to express my gratitude towards Zoltán Szlávik for reading and reviewing this thesis.

A lot of brainstorming occurred during the entire internship with other interns at Sentient, the importance of these sessions cannot be overlooked. I would like to thank Andrea Casati, Marijn Lems, Koen Pasman and again Jan Lasek for their creative input.

Last but definitely not least, I would like to thank my parents for their continuous support during my years at the university (and previous years).

Amsterdam, 31-10-2012
Vincent Hoekstra

Abstract

In this research a new type of prediction model is introduced, where an ensemble of k-nn prediction models is created with the help of an evolutionary algorithm. This approach is thought to be effective because of the importance of diversity in both ensemble models and evolutionary algorithms. This model consists of two stages, where the first tries to find those models that work well together within an ensemble and the second tries to learn the optimal weights of these individuals. Both of these stages are evolutionary algorithms, where the first stage uses deterministic crowding to find diverse prediction models. This model is then trained and tested by predicting games in the Dutch Eredivisie. The base two stage model is tested, together with some additional features. Of these features only weighted voting improved the performance of the model. The effect of the size of the ensemble and the number of variables on the performance was also tested. The ensemble created by the first stage is consistently better than that created by the second stage and these predictions are comparable but slightly worse than that of the best benchmarks. However, another ensemble model trained on the same dataset was beaten by the evolutionary ensemble model.

Keywords: *evolutionary algorithms, genetic algorithms, nearest neighbor, football, deterministic crowding, ensemble, prediction, bookmaker, weighted voting, diversity*

Abstract

In dit onderzoek wordt een nieuw voorspellingsmodel geïntroduceerd, waarbij een ensemble van k-nn voorspellingsmodellen wordt gecreëerd door middel van een evolutionair algoritme. De motivatie voor deze aanpak is dat bij zowel evolutionaire algoritmes als ensemble modellen de diversiteit van individuen een grote invloed heeft. Dit voorspellingsmodel bestaat uit twee onderdelen, waarbij het eerste onderdeel de verschillende individuele modellen probeert te vinden die goed met elkaar samen werken en het tweede onderdeel probeert de optimale gewichten te vinden voor deze individuen. Deze beide onderdelen zijn evolutionaire algoritmes, waarbij de eerste deterministic crowding gebruikt om verschillende individuele voorspellingsmodellen te vinden. Dit model wordt getraind en getest door middel van het voorspellen van voetbaluitslagen in de Nederlandse Eredivisie. Het standaard model wordt getest, samen met enkele toevoegingen. Van deze toevoegingen verbetert alleen weighted voting de kwaliteit van de voorspellingen. Daarnaast wordt ook het effect van zowel de grootte van de ensembles en het aantal variabelen getest. Het ensemble dat voortkomt uit de eerste stap van het totale model is consistent beter dan dat van het tweede onderdeel en de voorspellingen van het model zijn vergelijkbaar, maar net iets slechter, dan die van de beste benchmarks. Maar daarentegen doet het model wel betere voorspellingen dan een andere ensemble techniek getraind met dezelfde dataset.

Keywords: *evolutionaire algoritmes, genetische algoritmes, nearest neighbor, voetbal, deterministic crowding, ensemble, voorspelling, bookmaker, diversiteit*

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem description

In data mining, ensemble models are powerful prediction tools. Ensemble models make a decision based upon multiple underlying prediction models that work together as a team. Already in 1997, Thomas Dietterich stated that ensemble models would be one of the four main directions for data mining, when he showed that these models outperform the single classifiers that they consist of [9]. An indication of the power of these kind of models was given by the teams that won the NetFlix grand prize, when they used an ensemble model to win a price of a million dollars by predicting the preferences of people who watch films, [51]. This research will capitalize on the strong advantages of ensemble models and will do so in a novel and intuitive way.

In traditional ensemble methods, like boosting, individual predictors are trained separately. Instead, during this research base classifiers are developed simultaneously with the help of an evolutionary algorithm. This approach is assumed to be effective, because there is a key factor that determines both the success of evolutionary algorithms and the success of ensemble models. This crucial success factor is the diversity of individuals.

The power of ensemble models lies in the fact that uncorrelated errors between their members can be eliminated by using individual classifiers that are both as good and as different as possible. Likewise, evolutionary algorithms need population diversity so that they do not get stuck at less optimal solutions. This similarity is a fundamental motivation for this research.

Sentient's main application DataDetective has a fuzzy matching system, that finds those instances in a dataset that deviate the least from certain criteria. The core of this mechanism is a nearest neighbor algorithm. Sen-

tient is interested in using this core algorithm for prediction tasks. Since DataDetective is used in combination with a lot of different datasets and end-users are not necessarily data mining experts, it is important that such a prediction functionality is able to make robust predictions without much preprocessing or without much prior information about the dataset. The evolutionary ensemble method was chosen because it is thought that such a model can provide these robust predictions.

A suitable problem to test this evolutionary model on, is the prediction of football matches. Football is the most popular sport in the world and a lot of betting agencies exist that want to profit from this popularity. Bookmakers make predictions on football matches and allow people to wager against those odds. Being able to make better predictions than bookmakers should result in a profit in the long run. Besides the practical value of being able to predict football matches, there is another reason why football is suitable to test such a model. The outcome of a football match is very hard to predict, the few amount goals is one of the reasons why a lot of surprise outcomes occur. Predicting football is a hard problem, that will provide a challenge to any data mining application.

1.2 Goal of the thesis

The goal of the thesis is to create a robust prediction model based on an ensemble predictor that is trained with an evolutionary algorithm. This prediction model will be applied to the football domain, it should be able to predict the outcome of football matches. Key aspect here is that this prediction model should be able to make robust predictions without too much preprocessing or focus on parameter settings. Therefore, while this model is trained on football data, it should be able to be applied in any other field relatively easy. The goal is to investigate how this type of modeling can get good and robust results, rather than focusing on how to beat the bookmaker predictions or making a profit from bets. It has to be stated that in the original plan of this research the goal was to add sentiment data to the dataset and investigate how useful this would be for prediction. This way, a diverse dataset could be created where this type of modeling could be even more effective. This research was later dropped in favor of the research into ensemble models, because of time limitations and the fact that preliminary analysis by Sentient showed that public sentiment is not very useful for the prediction of football matches.

1.3 Structure of the thesis

This thesis is structured in the following way. Chapter 2 will start with an explanation of the domain in which this research takes place. The main aim of this chapter is to bring the reader up to speed concerning the three most important factors of this research, which are ensemble models, evolutionary algorithms and football prediction. Chapter 3 focuses on the dataset that was constructed for this research. Chapter 4 explains the base algorithm that is used for creating the ensemble prediction model. Chapter 5 explains and motivates how the evaluation of the models is done. Chapter 6 describes the setup of the experiments that were performed during this research, while chapter 7 presents the results of these experiments. Chapter 8 will then discuss these results and chapter 9 will conclude the thesis.

Chapter 2

Background & Related work

This chapter will be used to describe the domains that are relevant to this research. It will start by describing current models that are used to create ensembles and why ensemble models are so effective. Secondly, the general structure of the evolutionary algorithm will be explained. Finally, the football domain will be discussed. This includes which models are used in the literature to predict the outcome of games and how bookmakers profit from these predictions.

2.1 Ensemble models

An ensemble model is a model that creates multiple single prediction models and then aggregates these predictions somehow into a single prediction. Ensemble techniques are known to perform well on a variety of problems, and they tend to outperform the single classifiers that they are constructed from. The reason for this is given by [9], independent errors on the test set are diversified when using an ensemble. A prerequisite for this behavior is that the individual members in the ensemble are diverse, which means that they are able to produce better predictions on different test instances.

There are multiple ways of creating an ensemble classifier. In [5], Brown et. al. give a taxonomy of diversity creation in ensembles. In the paper, diversity creating methods are separated according to two distinctions. The first distinction is between *implicit* and *explicit* methods. Explicit methods use or try to optimise some measure of diversity when the ensemble is constructed, implicit methods create diversity in another way, without using a clear diversity measure. The second distinction made by the author is a lower level distinction between methods. The underlying models in the ensemble can be diverse by involving:

- *A different starting point.*
- *A different set of accessible models.*
- *A different way of optimizing the model.*

The first ensemble method is particulary useful with unstable algorithms. Unstable algorithms are algorithms that produce different results on multiple runs, often when slightly different training data is used. Examples of unstable algorithms are decision tree induction, [54] *page 317,* and Genetic programming [24]. In general, unstable algorithms are quite suitable for ensemble strategies. In [24], Johansson et. al. create ensembles of nearest neighbor algorithms with the help of genetic programming. The main focus in that research was to create ensembles based on this instability of the underlying models.

Some of the more popular ensemble creating strategies are the following algorithms.

- *Bagging*
- *Boosting*
- *Random Forest*
- *Stacking*

Bagging, or *B*ootstrap *Agg*regat*ing* [3], is an implicit ensemble method that uses a different starting point for constructing each ensemble member. For each individual predictor in the ensemble, a bootstrap selection is performed on the dataset. When applying one iteration of a bootstrap selection, the training data of the current model is constructed by sampling from the dataset until a training set is constructed that is the same size as the original data set. Because sampling is done with replacement, duplicates can occur in this training set. On average, the number of distinct instances in such a bootstrap training set is 63% of the original dataset. Those instances that are not in the training set will encompass the test set for that iteration. This way a different dataset is used for each iteration of the *Bagging* algorithm and each of these iterations produces one individual model for the ensemble. The final prediction of the ensemble is done by majority voting among the individual predictors.

Boosting is a similar technique, in that it is an implicit ensemble method using a different starting point for each creation of an individual model. But where in *Bagging* the central theme is the bootstrap selection, in *Boosting*

Algorithm 2.1 Boosting Algorithm.

Given a dataset of size N and k iterations (individual prediction models).
1. **Initialize** the weights at *1/N*
2. **For** *1 : k*
3. Create a bootstrap sample using the weights as a probability distribution.
4. train and test the model given de bootstrap training and test set.
5. calculate the error measure for this base classifier.
6. recompute the weights.

end for

7. Aggregate the predictions of the classifiers *1 : k* to get an ensemble predicion.

the key feature is that the relative importance of instances in the dataset changes. Usually, for the first iteration each instance in the dataset has the same weight, but in following iterations the weights of misclassified instances increase and the weights of correctly classified instances decreases. These weights can then be used for two purposes [49] *page 286*:

- They can be used as a sampling distribution for the bootstrap selection for the training set.
- They can be used by the base classifier for a weighted error measure on the test set.

The *Boosting* algorithm is specified in Algorithm 2.1. Aside from how the weights are used, different *Boosting* algorithms differ in the way the weights are recomputed and in the way the predictions are combined into an ensemble, [49] *page 288*. The most popular boosting algorithm is AdaBoost [12]. Boosting is effective because it forces underlying models to specialize on different instances in the training set, which leads immediately to a diverse ensemble.

Random Forests [4] are ensembles constructed from multiple decision trees. A *Random Forest* is a typical example of an ensemble method where the amount of accessible models is limited for each individual classifier. For each base classifier, again a bootstrap sample of the dataset is taken. Then for each node in the tree, a variable is necessary to split the node. This variable is selected by taking a random sample from all the available variables and choosing the best variable to split the node, among those that are in the sample. This imposes a limit on the way the tree can be constructed. Generally, trees in a *Random Forest* are not pruned, this creates even more

diverse trees. A limit on the number of nodes or a limit on the depth of trees is used instead.

Stacking, [55], is a different model compared to the others, in the sense that it does not necessarily construct the base models itself. Instead, stacking might perform the best given a set of predictors that are entirely different models. *Stacking* is performed by first training multiple base models on the dataset and then adding these predictions to the original dataset. Other models can then try to make predictions based on this improved dataset. This procedure can be repeated multiple times.

Aside from these popular models, some research has been done in creating ensembles with evolution. In [48], Sylvester and Chawla, create ensembles of neural networks with the help of evolutionary algorithms. However the research described in [48] is different compared to the current research because, in [48] individual classifiers within the ensemble are not created by the evolutionary algorithm but only the weights of already existing classifiers are optimized with an evolutionary algorithm. In [13], Gagné et. al. provide another way of creating ensembles with evolution, where they incorporate a diversity measure directly in the fitness evaluation.

2.2 Evolutionary algorithms

Evolutionary algorithms, First introduced by Holland [23], are a special kind of trial-and-error problem solving algorithms. They use the concepts that are characteristic for biological evolution, namely natural selection and recombination of genetic material [46], but apply it to problem solving. To explain how evolutionary computing works, it is best to start with explaining biological evolution.

In nature, species compete for the limited amount of resources that are available. The individual that is able to compete for these resources the best, has the biggest chance for survival, and therefore has a bigger chance of producing offspring. Due to genetics, this individual can pass along those specific traits, that made him the fittest member of the species, to his offspring. This mechanism ensures that entire species become generally more "fit" as generations continue. An important factor is that two members of a certain species combine their traits in their offspring. This creates entirely new members in the population, which keeps the population diverse.

This scheme can be translated into a problem solving algorithm. Instead of species competing for resources, individuals in a population depict solutions to a problem. The fitness of individuals of this population is determined by how good they solve the particular problem. Their chances of

advancing to the next generation depends on their fitness level. To keep the population from becoming exterminated, individuals should be able to combine themselve into offspring. This leads to one of the most challenging problems when designing an evolutionary algorithm. How should candidate solutions be able to pass along their traits to their offspring?

Each individual solution in the population is made up by two representations. The *phenotype* representation is the actual solution which is tested on the problem. The *genotype* representation is the equivalent of the chromosome of the solution, it is a piece of code that describes the phenotype of the solution. Designing a good genotype representation of individual solutions is the biggest challenge when applying an evolutionary algorithm. Another challenge is how to combine two parent genotypes to create offspring that have similar characteristics as their parents. How to design this genome and this recombination is often very specific to the problem that needs to be solved by the evolutionary algorithm.

The main scheme of an evolutionary algorithm is given by Algorithm 2.2. This scheme is roughly the same for each evolutionary algorithm. This general scheme immediately leads to those features of an evolutionary algorithm that still need to be designed. These are:

- Genotype representation - *The encoding of the solution, used for recombination.*
- Recombination - *The operator that defines how parent genotypes are combined into offspring.*
- Mutation - *The operator that defines changes on a single genotype when it is created.*
- Parent selection - *The operator that decides which individuals in the population are selected for creating offspring.*
- Survivor selection - *The operator that decides which individuals survive to the next generation.*

In general, an evolutionary algorithm can be described by describing the above features and applying algorithm 2.2.

The *recombination* and *mutation* operators have different roles within the evolutionary algorithm. The goal of the recombination is to take big leaps through the solution space while mutation is used for the finer search through the solution space. The *selection* operators are used to steer the population in the direction of fitter solutions.

Algorithm 2.2 General evolutionary algorithm scheme. *Source: [11], page 17*

BEGIN
INITIALIZE population with random candidate solutions;
EVALUATE each candidate;
REPEAT UNTIL (*TERMINATION CONDITION* is satisfied) **DO**
1. *SELECT* parents;
2. *RECOMBINE* pairs of parents;
3. *MUTATE* the resulting offspring;
4. *EVALUATE* new candidates;
5. *SELECT* individuals for the next generation;

OD
END

Normally, due to genetic drift [44], evolutionary algorithms tend to converge to just one optimal solution. But, as mentioned before, diversity is very important when considering evolutionary algorithms. The quality of the search that the algorithm performs depends on the amount of diversity within the population. The more diverse the population is, the more solutions are represented in the population. Because of this relation, a lot of evolutionary algorithms have been designed that try to maintain as much diversity within their population as possible. A special class of these, is the class of multi-modal evolutionary algorithms. With this type of algorithm, the goal of an evolutionary algorithm is not to find just the global optimum, but to find all local optima. When such an algorithm is applied, individuals in a population converge to different optima in the search space. Some popular multi-modal evolutionary algorithms are:

- *Clearing* [38, 44]
- *Crowding* [35]
- *Species conserving genetic algorithm.* [31]
- *Restricted tournament selection* [19]

In [44], the authors compare the above and other multi-modal evolutionary algorithms extensively. They conclude that among the best models are restricted tournament selection, a modified clearing approach, and deterministic crowding. These models are able to find most of the peaks in the fitness landscape. The difference between them is that the modified clearing approach could find all peaks, but was slow. Deterministic crowding found

less peaks compared to the modified clearing approach, but needed the least amount of computation time.

2.3 Football

2.3.1 Football betting

Bookmakers are companies who offer bets to customers. In sports, bookmakers offer bets on certain events during a game. This can range from the most common bet on the outcome of the game, to the number of yellow cards in a football match or the first player to perform a corner kick. The outcomes of these bets are highly unpredictable and this section will give a short description on how bookmakers make a profit on these kind of bets, despite the uncertainty.

At the heart of every bet lies a prediction made by the bookmaker. A bookmaker assigns a certain probability to every possible outcome of the bet. If a bookmaker is not interested in making a profit, it would allow people to make bets against these probabilities. Consider the example that people could bet on the match of Manchester United against Ajax and the bookmaker assigns probabilities of 75% and 10% respectively for the team to win. The profit that should be gained on a successful bet on these outcomes should be respectively 1.33 and 10 times the initial money at stake. These numbers are called the *odds* of a bet and when the bookmaker does not use a profit margin, these are called the *fair odds*.

Bookmakers like to make a profit and that is why they usually use a *profit margin*, this is a slight decrease in all the odds so that the pay offs are always slightly lower. This way, the expected value of a bet, based on the bookmaker predictions, is always slightly lower than the initial outlay. Aside from this mechanism, bookmakers tend to increase and decrease the odds of a particular outcome based on past betting behavior. Consider the previous example of Manchester United versus Ajax, as soon as a lot of people start to bet on Ajax, the bookmaker will start to decrease the odds of this outcome en slightly increase the odds of the other outcomes. This way, a bookmaker diversifies the risk and this almost always guarantees a profit roughly equal to the profit margin the bookmaker uses.

Coming up with a profitable betting strategy against bookmakers is a difficult task. The most common approach is that of *value betting*, which means betting on those outcomes where the probability estimated by the bookmaker is lower than predicted by the person who wants to place a bet. This essentially means that the better thinks that the bookmaker made a

mistake in assessing the probabilities of the outcome of a game.

Coming up with a viable betting strategy comes down to one of two options. The first one is to be able to make better predictions than the bookmaker, however, if this is the case than becoming a bookmaker might be a more profitable strategy. The other option is to spot those games and those outcomes of games where bookmakers tend to make mistakes and bet on those outcomes if such a mistake is perceived.

2.3.2 Predicting football

Match outcome predictions have been studied for multiple decades now. Maher [34] was one of the first to use a statistical approach to model the outcome of a football match. He modeled the amount of goals scored by both teams with a Poisson distribution. In [10, 41], the authors provide extensions to this model, so that it can be used for prediction.

The previous method predicts the amount of goals for each team and infers the actual winner of the game from this result. A more recent approach is to determine the outcome (win, draw, loss) of the game directly. Koning, [28], used this in his research about balance in the Dutch national football league and Goddard, [15], implemented a similar model in a comparative study. Both use an ordered regression model to predict the outcome of the game. According to Goddard, Poisson models and ordered regression models are the most used methods for predicting football, in his comparative research he concludes that these methods have a very similar performance. Both types of models use a similar approach, in that they use ratings to measure the current strength of both of the teams and predict the result based on this difference in strength. These ratings might be influenced by home advantage or, for example, the fact that the team can still win the competition and is therefore more motivated. In these models, the ratings of both teams are compared and predictions are inferred from this comparison.

In other studies about football prediction, Leitner et. al. [30] try to predict the outcome of the 2008 European championship based on Elo ratings and bookmaker odds. Luckner et. al. [33], try to predict football outcomes with the help of prediction markets. An artificial Intelligence technique was used by Constantinou et. al. [8] in the form of a Bayesian network, when the authors tried to incorporate expert opinions in their model. Evolutionary algorithms where also used for football predictions on few occasions. Rotshtein et. al. [39] used a genetic algorithm and fuzzy model to predict the outcome of games. Rowan, [40], evolved betting strategies for betting on the outcome of football games, and was able to get a profit with this model.

Chapter 3

Data Description

This chapter will explain how the data was retrieved and how the dataset was created that was used during this research.

3.1 Data retrieval

Data about football is scattered all over the internet. However, *football-data.co.uk* provides a complete dataset about fixtures and betting odds that were invaluable during this research. It enabled variables that reflected past performance and provided a clear benchmark to the problem, the predictions based on bookmaker odds. Other data had to be retrieved from other sources. In [16], Goddard used multiple variables for the prediction of football matches. Among these variables were: whether or not one of the teams was eliminated from the cup, and the distance between stadiums. Aside from these variables, each pairing of team i and team j has been given a rivalry score, depicting the rivalry of team i towards team j. This way each match has two rivalry scores.

Sources:

- League data: *football-data.co.uk*
- Betting odds: *football-data.co.uk*
- European cup games: *statto.com*
- Domestic cup: *fcupdate.nl*
- Stadium distance: *Google maps*

The data was retrieved with Selenium, which can be found here: *http://seleniumhq.org*.

3.2 Data preparation

After the data is retrieved, it needs to be processed to a format suitable for nearest neighbor algorithms. Since individual matches need to be predicted, each instance in the dataset should depict one match. This way, a match can be predicted by looking at the most similar instances. The distance between different football matches is then calculated by comparing variables within the dataset. The main topic of this section is how these variables are constructed.

To start, each match consists of two teams competing at the grounds of one both teams. This way, each match in the dataset has a hometeam and an awayteam, both have scored an amount of goals from which the result of the match is derived: {Home win (H), Away win (A), Draw (D)}. This result is the variable that has to be predicted by the algorithm, for a given football match.

The goal here is to create a vast dataset with a lot of variables, the selection of the useful variables for prediction is done by the evolutionary algorithm. Therefore, no prior knowledge is necessary about the usefulness of these variables. Variables can roughly be separated in two categories: Variables that reflect *past performance of teams* and *miscellaneous* variables. In the dataset that is used in this research, the variables that do not depict past performances are:

- Team names.
- Rivalry from one of the teams to the other.
- Whether a team plays in a cup next week.
- whether a team played in a cup last week.
- Geographical distance between stadiums.

This is a short list, while the original intention was to create a diverse dataset about football, this proved to be harder than expected. Lots of data is available about football, for example grades given by experts to players, information about the lineups or sentiment data. However, this data only goes back a small period of time. Most information is only stored for the current season. This posed a lot of restrictions on the type of data that could be used during this research.

Because of this reason, most of the variables in the dataset are variables that reflect the past performance of one of the teams, or the relative performance of one team compared to the other. These are variables that

depict, for example, the number of goals or the number of points scored by a particular team. When considering such variables, some *dimensions* play an important role. These dimensions are characteristics about a variable, where each variable must be about:

- A team - *Home team, away team, or the difference between both.*
- A location - *At home, away, or in total.*
- A time interval - *For example, the last 1,3,10, or 34 games.*
- A quantity - *Points, goals, goals conceded, or goal difference.*

All the above distinctions are important when considering football, information about both of the teams is relevant, but the difference in past performance of both teams might be more related to the actual chances of winning. A reason why the information about the individual teams is relevant as well, is that if, for example, both teams scored a lot of goals in recent matches, the difference between them is still around zero. However, the probability of a draw happening is decreased in this scenario, because goals are likely to be scored and the probability of a draw decreases when a lot of goals are expected. Whether or not the goals where scored at home, away or in total is also relevant because some teams have a bigger advantage of playing at home and this should be reflected in past performances at their home ground. Whether or not these performances are long term or short term might also make a difference, that is why a time frame is considered. For this research, looking back 1, 3, 10 and 34 matches was considered a good selection to capture any short term and long term behavior.

There are a couple of variables in the dataset that are obtained with an ordered regression model [15, 28]. With this model, a rating can be given to both the home and the away team. This rating depicts the current strength of the team and outcome probabilities can be inferred from these ratings. In this dataset, ratings were constructed by using: all the previous matches, the last 30 matches, or the last 10 matches. these three sets of ratings were used to calculate outcome probabilities, which were also put in the dataset. These ratings were provided by Sentient. For more information about them see [29].

When combining all the possibilities for the dimensions, mentioned earlier, there are a lot of variables to consider. For example one of these variables could be the points scored by the home team at home and away in the last 3 games. The total amount of variables that can be created from these combinations are 144. However, a few were dropped because they are not

considered to be relevant. These variables were variables that computed the difference in performance of the away team while playing at home and the home team playing away.

The dataset consists of seven seasons, because some variables could only be retrieved for that time span. From this set of matches, the first two weeks had missing data, so they were removed as well. The total dataset is comprised of 2124 matches with a total of 149 variables.

Chapter 4

Model description

This section will describe the basic version of the evolutionary ensemble model that has been developed during this research. It will start with a basic description of the underlying nearest neighbor mechanic and explains the evolutionary model afterwards. The training of the model has been split into two stages, the models for both stages will be explained in greater detail.

4.1 k-nearest neighbor

Nearest neighbor algorithms belong to the lazy class of models. A lazy model is not trained on training data, but classification is done by modeling the training data each time a classification needs to be made, [49] *page 223*. Because there is no model trained beforehand, no computation time is necessary at that time to compute such a model. Instead, computing predictions for a test instance can become quite expensive, because all the distances between the test instance and all the training instances need to be calculated.

Nearest neighbor algorithms need an extensive amount of preprocessing before they can be used. For example, all numeric values should be normalized to assure that all variables are equally important. Fortunately, in this research this normalization and preprocessing is managed by DataDetective. Within the DataDetective framework, given a dataset, it is enough to provide a new instance and the variables that need to be considered for the matching algorithm and DataDetective will return the k nearest neighbors according to those variables. This is very useful when considering an evolutionary algorithm, because the only thing that separates two different nearest neighbor algorithms from each other is which variables are used for

prediction.

These k neighbors are used to make a prediction. The match outcome of all these neighbors is known. To give a prediction for the match that needs to be predicted, a probability distribution can easily be derived by taking the frequency of each outcome among the neighbors and dividing this number by k.

4.2 Two-stage model

The evolutionary ensemble model consists of two stages, where the result of the first stage is used by the second stage. Both of these stages are evolutionary algorithms and both are able to provide an ensemble prediction model based on individual k-nn predictors. The first stage is the most important stage, this is the stage where the individual k-nn predictors are found that are useful when working together. Logically, the individuals in the population of this evolutionary algorithm are k-nn predictors. The final population of this stage is provided as input for the second stage.

The second stage tries to find the optimal ensemble, given the individual predictors provided by the first stage. Individuals in the population of this evolutionary algorithm consist of ensembles, where each base k-nn predictor in such an ensemble has a weight that determines the relative importance of that predictor within the ensemble. The last population of this second stage is used to provide a prediction based on the optimized ensembles.

The main difference between the result provided by the first stage and second stage is that the first stage simply uses all the k-nn models that are provided with equal weight. The second stage, on the other hand, makes a selection from all the classifiers and provides prediction based on a weighted average on these selected individual k-nn models.

4.3 First stage

The goal of this stage is to find as much diverse predictors as possible. As stated in chapter 2, there is a specific class of evolutionary algorithms that deals with this type of problems where multiple optima need to be found. These multi-modal evolutionary algorithms are ideal when creating base classifiers for an ensemble model. Deterministic crowding is chosen as a suitable algorithm for this problem, because of its speed and performance, [44].

One iteration of the deterministic crowding algorithm is specified in al-

Algorithm 4.1 Deterministic crowding, source: [44]

1. Select two parents, p_1 and p_2, randomly with no replacement.
2. Perform crossover between them , yielding c_1 and c_2.
3. Apply mutation/other operators, yielding c_1' and c_2'.
4. if $[d(p_1, c_1^{'}) + d(p_2, c_2^{'}) \leq d(p_1, c_2^{'}) + d(p_2, c_1^{'})]$

- if $f(c_1^{'}) \geq f(p_1)$ replace p_1 by c_1'
- if $f(c_2^{'}) \geq f(p_2)$ replace p_2 by c_2'

else

- if $f(c_2^{'}) \geq f(p_1)$ replace p_1 by c_2'
- if $f(c_1^{'}) \geq f(p_2)$ replace p_2 by c_1'

gorithm 4.1. During each generation of the evolutionary algorithm, this algorithm is applied until each member in the population is used exactly once for recombination. This ensemble creation scheme creates diversity in an explicit way, since deterministic crowding needs an explicit diversity measure, which is specified by the value *d(a,b)* in the algorithm.

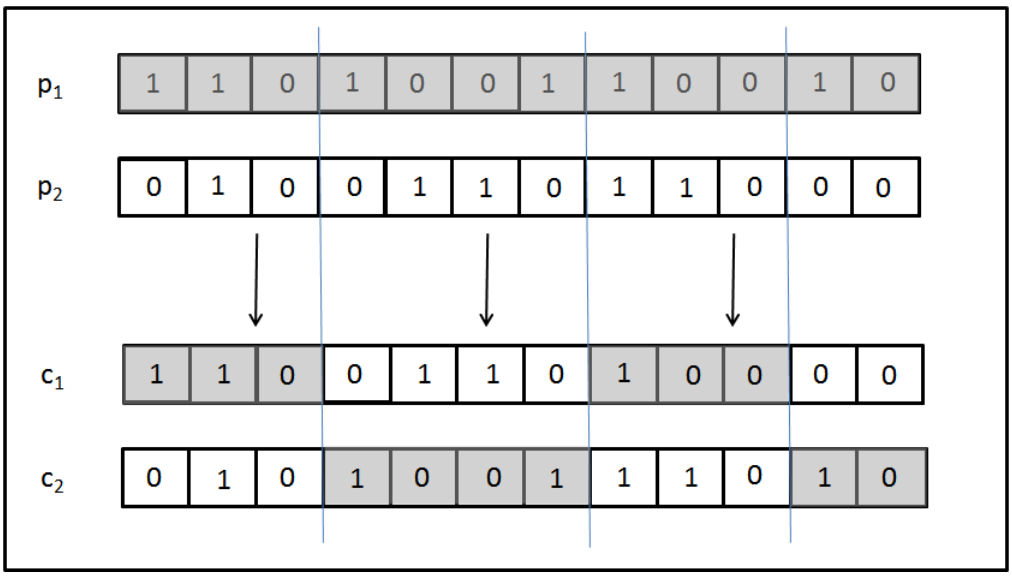
The deterministic crowding algorithm is first introduced in [35], and it ensures population diversity by the fact that new members are only added to the population if they have a better fitness than their most similar parent. The evolutionary operators, for example for parent recombination, are not specified by the algorithm. These modeling choices for the evolutionary operators for the first stage are specified in table 4.1 and motivated below.

Operator	Description
Phenotype	k-nearest neighbor classifier
Genotype encoding	Bit string
Recombination	3-point crossover
mutation	bit flipping
selection	Based on deterministic crowding
diversity measure	Hamming distance

Table 4.1: Operators for the first stage of the algorithm.

Genotype encoding – each nearest neighbor model only differs in which variables it uses for the prediction, for the encoding it is enough to specify this difference. The encoding of a model consists of a bit (either 0 or 1) for every variable, where a '1' means that the variable is used to calculate the difference to neighbors in the training set. The encoding of one individual prediction model then becomes a bit string. The motivation behind this approach is that there is a structure in the ordering of the variables in

Figure 4.1: 3-point crossover.



the dataset and by using a bit string, the genome takes this ordering into account. For example a part of the genome could be 01110, meaning that the three variables in the middle are switched on because they might work good together.

N-point crossover is chosen for the *recombination* of these individuals. N-point crossover is done by taking the two parents and selecting n points randomly in the genome. All of the segments between those points are either given to child 1 or to child 2. The number of crossover points that is used by the algorithm is three. Picture 4.1 shows an example of 3-point crossover.

The motivation for 3-point crossover is that many variables in the encoding are somehow linked to adjacent variables. For example "goals scored by the away team" and "goals conceded by the away team" could be stored next to each other in the genome. 3-point crossover ensures that these structural properties are not constantly broken up during crossover.

Mutation is done by simply flipping a bit from '0' to '1' or vice versa. Each bit in the genome has a probability p, which is the probability that the bit will flip when it is created. This probability is set so that on average one bit will be mutated for every creation of a child.

Deterministic crowding also needs a *diversity measure* to check which parent should be compared to which child. The Hamming distance is used as a measure for the diversity, which is simply the number of bits in the string that are not equal. When considering the phenotype representation,

Operator	Description
Phenotype	Ensemble of classifiers
Genotype	Vector of indexes and weights
Recombination	Uniform
Mutation	Random Resetting
Parent Selection	Ranking based selection
Survivor Selection	Tournament selection

Table 4.2: Evolutionary operators for the second stage.

the Hamming distance describes the amount of variables that are not shared by both individuals.

4.3.1 Initialization

The initialization of the algorithm deserves special attention. Initialization in evolutionary algorithms is generally random to create a diverse initial population. The most straightforward way to create randomness with this encoding is to have a 50/50 chance for each bit to be either a 1 or a 0. However, this results in a population where each individual had around 50% of its variables switched on at initialization. This situation is unwanted, because it is not sure if this is the best amount of variables to use. By changing the probability of a 1 to occur at initialization, the average number of variables switched on can be manipulated. This way, multiple tests can be performed to check what kind number of variables performs the best.

4.4 Second stage

For the second stage, the goal is to use the individual models provided by the first stage and combine some of these in an ensemble that should be able to predict the outcome of football matches better than the single classifiers. Table 4.2 shows the evolutionary operators for this part of the algorithm.

Each of these ensembles consists of a predetermined number N of single predictors. These N classifiers are chosen from those given by the first stage. Each of these classifiers within the ensemble has a weight. This weight determines how heavy the prediction of this individual counts compared to the others in the ensemble, these weights sum up to 1 for convenience. Each of the classifiers returned by the first stage gets a unique index number. The *genotype* then consists of N pairings of this index number and a weight. The motivation why this method is preferred over simply using all the available

k-nn predictors for each ensemble is that deterministic crowding divides its population evenly over all the peaks in the fitness landscape. This means that, while the entire population is diverse, there are some similar individuals in the population of the first stage. By taking a maximum of N predictors, these redundant prediction models should be eliminated from the ensemble. Another reason is that it allows more control over the algorithm, to check which ensemble size performs the best.

The *recombination* operator is explained in picture 4.2, it resembles uniform crossover, where each pairing, of a weight and index, in the genotype has a 50% chance to belong to child1 or child2. For *mutation*, each index has a chance of resetting on creation, which means that the index changes to another that was not yet in the ensemble. Each weight has a probability of changing, when this happens, the weight resets and all the weights are normalized again. *Parent selection* is done by favoring individuals that have a higher fitness ranking. Ranking based selection is preferred over the most well-known fitness proportionate selection. Fitness proportionate selection tends to converge faster, and therefore has a higher risk of premature convergence. Ranking based selection is done based on formula 4.1, *source:* [11]. In formula 4.1, i is the rank of the individual, given that the fittest individual has rank $\mu - 1$ and the lowest has rank 0. S is a parameter that specifies the skewness of the distribution towards the fittest member. This parameter should be between 1.0 and 2.0 and in this implementation 1.6 is chosen. The number of children created is kept at 50% of the total population size. Survivor selection is then performed by tournament selection [11]. The size of the tournaments is 20% of the total population size.

$$p_{selection}(i) = \frac{2-s}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)} \tag{4.1}$$

The final prediction of the second stage is given by averaging all the predictions of the ensembles in the final population.

Figure 4.2: Recombination operator for the second stage.

Chapter 5

Model evaluation

Correct performance evaluation is crucial in both data mining and evolutionary algorithms. In data mining the actual performance of a model can only be adequately tested when a test set is used that is not applied in any way during the training of the model, [54] *page 146*. When that model is an evolutionary algorithm this issue is even more important, because evaluations occur during the learning process in the form of fitness evaluations. This chapter will specify two important choices regarding the model evaluation. The first part of the document will consider the choice for the training and test set. The second part will discuss the choice for an appropriate performance measure for a football result prediction model.

5.1 Choice of training and test set

The dataset consists of seven seasons, all of which contain 306 matches. This is divided into a *training set,* a *fitness calculation set*, and a set to estimate the actual performance of the entire model, a *test set*. During a generation in the evolutionary process, fitness scores are used to determine survival chances. To make sure that differences in fitness between members in a population are reliable, the same test set is used for each individual in a population. This approach has a drawback, the evolutionary process will evaluate each generation on the same sequence of matches and will tend to overfit on those matches. Chapter 6 introduces a couple of strategies that try to counter this behavior. In the original model the dataset is split in the following way:

- Training set: *first 4 seasons.*
- Fitness evaluation set: *5th and 6th season.*

- Test set: *last season.*

The choice to make the fitness evaluation set bigger than the actual test set was because overfitting is done on this fitness evaluation set and it should be as big as possible to counter this overfitting problem. When the actual performance of the model is tested, all the previous instances are used as a training set, including the previous fitness calculation set. This ensures that matches are predicted by using the most recent information but never predict based on the future. Strategies like bootstrapping and cross validation were not used because matches would be predicted based on future data, which could be considered an unfair advantage.

5.2 Performance measures

The goal of the model is to come up with predictions that are better than those of the bookmakers. Therefore, a classification of the most likely result is not enough. Bookmakers give their predictions in the form of probability distributions. The prediction of the model has to be in the form of a probability distribution on the outcome of the game {*P(h), P(a), P(d)*} as well. Evaluating a predicted probability distribution on a given match is difficult, because the only information that can be used is the actual outcome of the game.

Intuitively, one of the first measures that comes to mind is the probability assigned to the actual outcome. However, that this measure is not appropriate can easily be shown by an example. If the correct probability distribution for a match is {0.5,0.3,0.2}, than the expected reward for predicting {1,0,0} is higher than for predicting the actual probability distribution. Using this measure is equivalent to predicting the most likely outcome, because on average this will score better.

There are, however, some measures that are more appropriate when trying to predict probabilities. These are the log-likelihood (or the equivalent information loss function), depicted in eq. 5.1 , the quadratic loss function, eq. 5.2, and the rank probability score in eq. 5.3.

$$log - likelihood = log_e(p_r) \tag{5.1}$$

$$quadratic\, loss = \sum_j (p_j - a_j)^2 \tag{5.2}$$

$$Rank\ Probability\ Score = \frac{1}{k-1} \sum_{j=1}^{k} (cdf(p_j) - cdf(a_j)^2 \tag{5.3}$$

Where p_j is the estimated probability of class j, a_j is the actual outcome of class j, which is 1 or 0, k is the number of classes, which is three, p_r is the probability that was attributed to the actual result, and *cdf* is the cumulative distribution function of that variable.

The first two are common approaches, [54] *section 5.6*, for predicting outcome probabilities, but the log-likelihood has the nice property that it is similar to the return from bets, [54], which is appropriate in this case. The rank probability score is suggested in [7], by Constantinou and Fenton, as a good alternative for the prediction of football outcomes. The authors state that other measures are not appropriate, because the results in football are from an ordinal scale. They state that predicting a draw, instead of an away win, should be rewarded more if a home win happens. In principle this seems correct, but it has been observed that for teams of similar strength, a draw is often the least likely outcome. Because of this reason, penalizing away win predictions in this case seems unfair. The rank probability score can be useful in another situation. For the fitness evaluation, it might be useful to predict the goal difference instead of the outcome of the game, this might result in some different nearest neighbor classifiers that can be used in the second stage of the algorithm. The rank probability score is suitable in this case, because the goal difference definitely has an ordinal scale.

Another option, since bookmaker data is available, would be to calculate the money gained from bets directly. The advantage here is that this gives a result that is easier to interpret than the measures mentioned above. There are, however, a couple of drawbacks. In [53], the author concludes that there is only a weak relation between predicting profit and predicting actual outcomes in finance. The best model to predict profit does not necessarily have to be the best prediction model. Aside from this, the two main reasons not to predict profit is that such a model is trained with the goal to find faults in the predicting system of a bookmaker, which might perform worse if the bookmaker starts to use a different model. The other reason is that profits depend a lot on the betting strategy that is applied, results might be more dependent on this strategy than on the actual prediction capabilities of the model.

To conclude, the model will predict matches based on a $\{P(h),P(a),P(d)\}$ probability distribution. The fitness of an individual in a population will be the average log-likelihood performance on the fitness calculation set, because the log-likelihood is an appropriate measure when considering bets. For the final result, not only the log-likelihood will be presented but also the percentage of correctly classified instance and the quadratic loss function. The percentage correct is useful to present because it is easy to interpret

and the quadratic loss function is added because it takes into account the entire outcome probability distribution, and might therefore give some extra information.

Chapter 6

Experimental setup

This chapter will describe the experiments that are performed to check the performance of the evolutionary ensemble model. The first section describes the setup for a test that should give an overview of the general performance of the base model, described in chapter 4. The second section of this chapter will describe tests that involve changes to this base model. The third part will describe a couple of tests that have been performed to check what the effect is of both the size of the ensemble and the number of variables used for prediction.

6.1 Base model performance

Evolutionary algorithms optimize a model based on an iterative procedure. To check whether this procedure is successful, it needs to be checked if the performance of the model increases as the number of iterations increases. This check will be performed by running the *first stage* of the base model for 36 generations, for each generation the following performance indicators will be presented:

- The average log-likelihood performance of the individuals in the population on the test set.
- The average log-likelihood performance of the individuals in the population on the training set.
- The log-likelihood performance of the ensemble, constructed from the individuals, on the test set.
- The log-likelihood performance of the ensemble, constructed from the individuals, on the training set.

Table 6.1: Feature test settings.

Description	Value
1st stage population size	250
1st stage # generations	10
2nd stage population size	700
2nd stage # generations	20
k	30
ensemble size	40
# k-nn variables at initialization	30

The goal of this experiment is to check whether the performance of the model increases during the evolutionary algorithm and to check how much the model overfits on the training data.

6.2 Model feature testing

The goal of this research was to create a model which is able to give robust predictions without the need for parameter optimization. For example, the value for k in k-nearest neighbor algorithms is considered an important variable. However, the value of k depends heavily on the context of the problem. For this reason, these tests are performed with a fixed value for k with no knowledge about the suitability of this value, simulating a situation where no information is known about the problem. The settings that were used during these tests are depicted in table 6.1.

Instead of finding those parameters that will beat the benchmarks, the focus of these experiments will be on testing multiple added features or changes to the algorithm that might change the behavior of the algorithm. The results of these tests will be compared to the results of the base model. The main two questions that have to be answered by these experiments are:

- How does the performance of the ensemble model change because of these added features?
- How robust is the model?

6.2.1 Goal difference

A fitness measure is often very problem specific and very important for the success of an evolutionary algorithm. It was decided in chapter 5 that the

fitness would be calculated based on the log-likelihood of the outcome probabilities, where these outcomes are home win, away win, or a draw. It would be interesting to see how the performance changes if a completely different fitness measure would be used to create individual ensemble members.

Instead of trying to predict the winner of the game, *the fitness of individuals* will be calculated based on how good they are able to predict the goal difference. The goal difference of a match ranges from -5 to 5, where extreme results are put in the class of -5 or 5 depending if the home team or the away team won. This prediction will then be rated by the rank probability score, eq. 5.3. The ensemble models that are created will be tested again by trying to predict the winner of the game instead of the goal difference. This will test how good the ensemble model is able to handle changes in the fitness evaluation procedure.

6.2.2 Prediction based diversity

To create diverse individuals during the first stage, deterministic crowding uses a diversity measure. The current diversity measure is the Hamming distance, which measures the absolute difference in variables used by individual k-nn classifiers. As was mentioned in chapter 2, the diversity in ensembles is measured by the difference in instances that can be predicted correctly by the individual predictors. Instead of using the Hamming distance, another diversity measure might be more appropriate. The difference in prediction can be measured by the correlation between predictions made by two individuals. The diversity measure suggested here is the inverse of the pearson-correlation coëfficient, based on the predictions made by two different classifiers. This should be more closely related to diversity within ensembles than the Hamming distance.

6.2.3 Different mutation operator

The current mutation operator for the first stage is bit-flipping, where each bit in the genotype representation has the same chance of flipping to the other value. When the ratio of ones compared to zeros starts to differ from 50/50, bit flipping will favor going back to a ratio of 50/50. In this case it might be preferable to change the flip-probabilities, so that the probability of mutating an extra one is equally likely as mutating an extra zero. This way, mutation in the genome means either increasing or decreasing the complexity of individual k-nn prediction model with equal probability, which seems more appropriate. This type of mutation is even more applicable when considering

the fact that individual k-nn prediction models are initialized with a specific expected number of ones in the genome.

6.2.4 Subset fitness evaluation

Currently, individuals within a population are tested on the same set of matches, this ensures a fair evolutionary process. The problem that occurs is that the individuals within this population tend to specialize on this entire fitness evaluation set. A solution to this problem might be to create an 'unfair' evolutionary process where, during each generation, each individual is tested on a different subset of matches. While this process might favor some individuals during one generation of the evolutionary algorithm, it might provide steady longterm results. In the long run, each individual in the population will be expected to have the same amount of 'luck'. This type of evolutionary algorithm mimics evolution in nature, where individuals also have a specific amount of luck during their lifetime. Since this scheme does not focus on a specific set of matches, it might provide more robust classifiers and result in less overfitting. This strategy will be used on both stages of the algorithm, since both have a tendency to overfit.

6.2.5 Different test set second stage

This feature introduces another approach to counter overfitting behavior. The current model trains both stages on the same fitness evaluation set. It might perform better if one season is used for evaluation during the first stage and another for evaluation during the second stage. This might result in more general models.

6.2.6 Weighted voting

In [56], Zavrel states that k-nearest neighbor algorithms with weighted voting are more robust regarding the choice for k, which means that weighted voting should provide better results if a less suitable amount of neighbors is chosen. In the base model, each of the neighbors has an equal vote in constructing the outcome probability distribution. With weighted voting, the influence of each neighbor on this probability distribution is weighted by the distance to the instance that needs to predicted. Since the goal is to create a robust algorithm, independent of the choice of k, weighted voting will be tested to see whether it has a better performance than the standard model.

6.3 Other tests

6.3.1 Initial number of variables

As stated in chapter 4, the model allows a control over the number of variables the individual predictors use at the initialization of the evolutionary algorithm. This test is designed to check what the effects are if this amount is varied. This gives useful information about the robustness of the algorithm. Moreover, reducing the amount of variables reduces the complexity of the model, which might result in less overfitting and a faster algorithm.

To perform these tests, all the features, mentioned in section 6.2, that perform better than the standard model will be applied. Then the amount of initial variables will be varied to get an understanding of the effects of the number of variables, used by single k-nn classifiers, in the model. Again, the goal of this experiment is to assess the quality and robustness of the ensemble model.

6.3.2 Ensemble size

It was explicitly chosen in chapter 4 to limit the ensemble size. This provides an opportunity to vary this size and check the effects of this change. A slightly different test was used for this test. To make the difference more clear between the different sizes of the ensemble, the second stage is run for 40 generations instead of 20, and the average performance is taken from all the ensembles in the final generation. This test is performed twice and the results of both tests are presented. The goal of this experiment is to check whether this bound to the ensemble size is a useful feature.

Chapter 7

Results

This chapter will present the results of the experiments that were performed. These results will provide a clear understanding of the performance of this type of prediction models on the football domain. Figure 7.1 shows the results of the base model. This figure shows four graphs, the performance of both the average individual models and the ensembles on both the test set and the training set. This graph shows the error measure based on the log-likelihood, which decreases for all four graphs. Figure REF shows only the performance of the ensembles on the test set, to get more detailed view of the performance.

Table 7.1 shows the results of the experiments that were explained in section 6.2. Each set of two rows in this table corresponds to one of the performed tests, the first row states the performance of the ensemble based on the results of the first stage and the second row states those of the second stage. For each of these tests one of the features was switched on. This table provides information about which features enhance the performance of the model. The best performance for each measure are presented in **bold**. The performance of the standard model is depicted in the first two rows of this table. The tables in this chapter provide the percentage of games that the correct outcome was predicted, the log-likelihood and the quadratic loss function. For the latter two 90% confidence intervals are also presented. It is important to note that for the percentage of correct outcomes the highest figure is the best, while for the other two the lowest is the best score.

Figure 7.1: Performance base model.

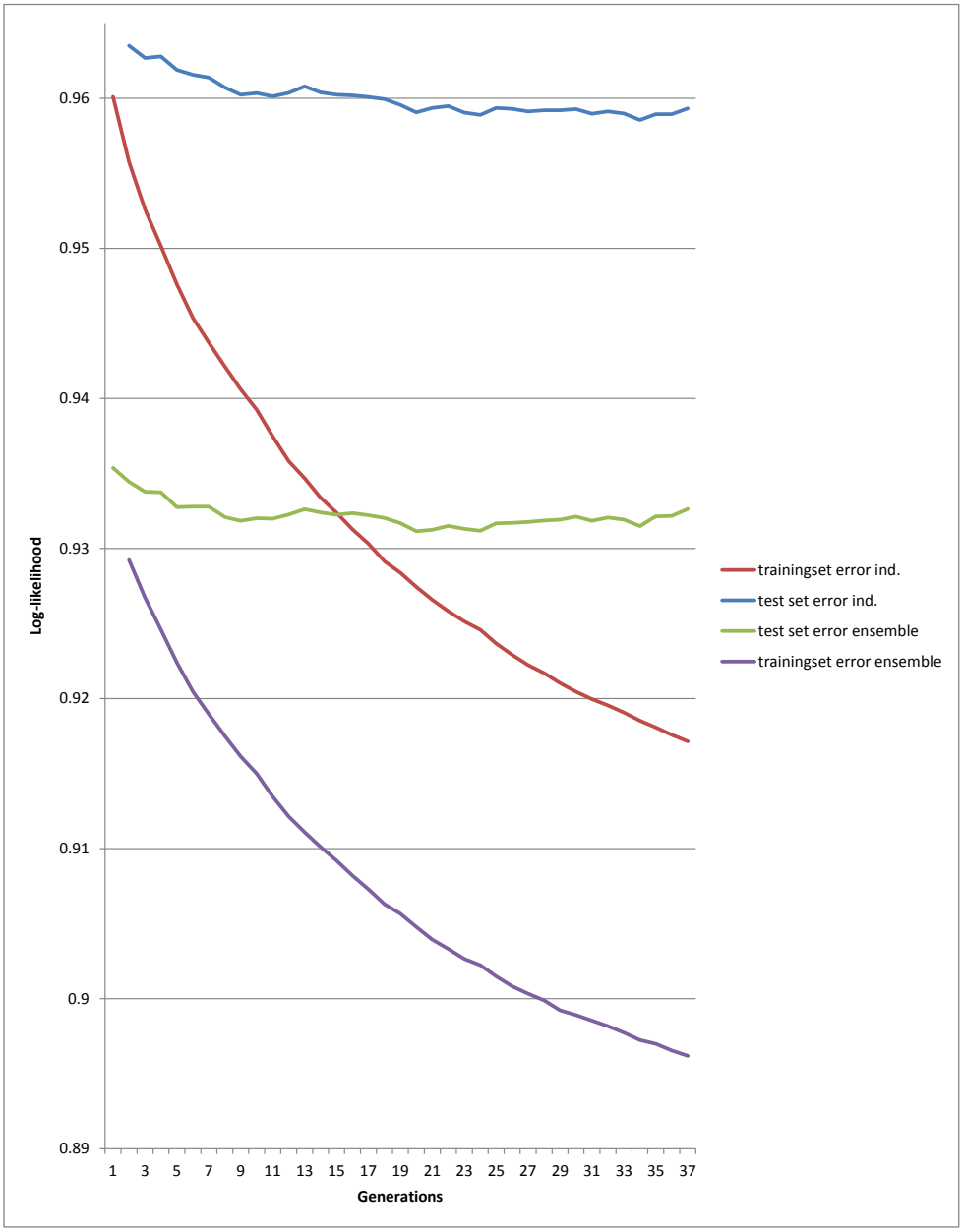


Figure 7.2: Ensemble performance.

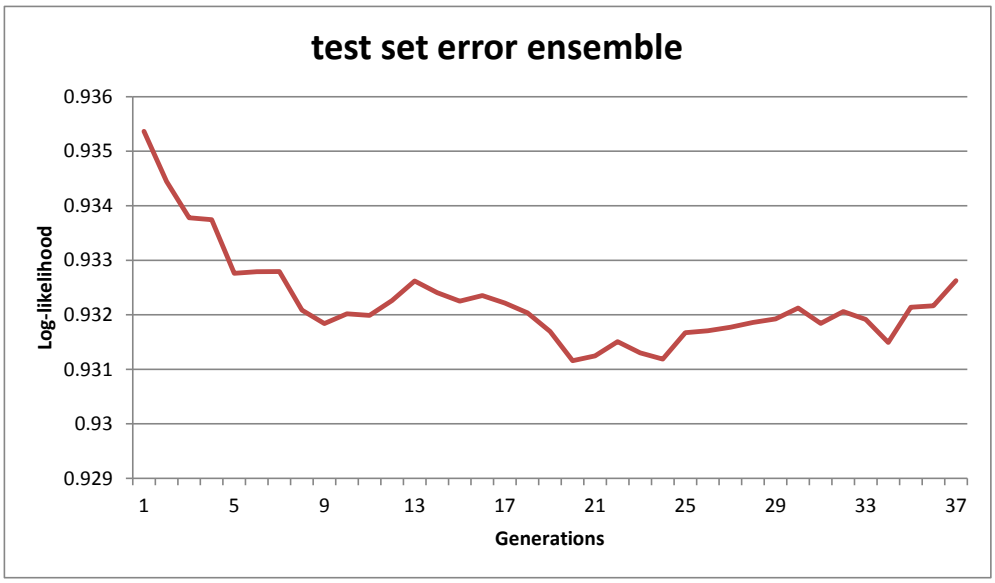


Table 7.1: Results algorithm features.

Feature	% correct outcome	Log-likelihood	Quadratic loss
standard - 1st stage	56.86	0.9328 (0.8881, 0.9774)	0.5521 (0.5207, 0.5835)
standard - 2nd stage	**57.52**	-0.9336 (-0.8885, -0.9788)	0.5528 (0.5211, 0.5845)
goal difference - 1st stage	56.21	0.9328 (0.8875, 0.9781)	**0.5517 (0.5200, 0.5834)**
goal difference - 2nd stage	56.54	0.9352 (0.8891, 0.9814)	0.5531 (0.5210, 0.5851)
modified mutation - 1st stage	55.88	0.9330 (0.8888, 0.9773)	0.5521 (0.5210, 0.5831)
modified mutation - 2nd stage	56.54	0.9339 (0.8891, 0.9788)	0.5529 (0.5214, 0.5845)
prediction correlation - 1st stage	55.56	0.9332 (0.8892, 0.9772)	0.5522 (0.5211, 0.5832)
prediction correlation - 2nd stage	56.54	0.9364 (0.8912, 0.9816)	0.5537 (0.5220, 0.5854)
weighted voting - 1st stage	56.54	**0.9315 (0.8870, 0.9761)**	0.5520 (0.5207, 0.5834)
weighted voting - 2nd stage	56.54	**0.9315 (0.8865, 0.9765)**	0.5519 (0.5203, 0.5835)
different set 2nd stage - 1st stage	55.56	0.9349 (0.8908, 0.9790)	0.5535 (0.5224, 0.5845)
different set 2nd stage - 2nd stage	56.21	0.9346 (0.8901, 0.9792)	0.5534 (0.5221, 0.5847)
subset fitness evaluation - 1st stage	55.56	0.9362 (0.8934, 0.9790)	0.5545 (0.5242, 0.5848)
subset fitness evaluation - 2nd stage	55.56	0.9371 (0.8943, 0.9798)	0.5550 (0.5248, 0.5853)

As explained in section 6.3.1, those features that perform better than the standard model are combined for the next test. The only feature that increased the performance of the base model was weighted voting. The tests that were performed with this model are shown in table 7.2. In this table, the results are shown of the tests where the number of variables is varied. This number of variables is the expected amount of variables used by the individual k-nn prediction models upon initialization of the evolutionary algorithm. These results can be compared to those of benchmarks in table 7.3. For an explanation of these benchmarks, see [29].

Figure 7.3 shows the average performance of the ensembles after 40 generations of the second stage. It shows the result for an ensemble size of 2 up until 49. Two tests where performed and both are shown in this figure.

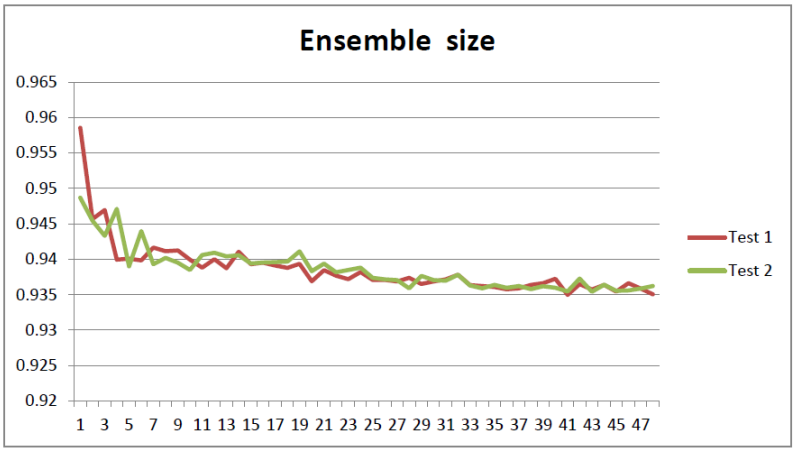
Table 7.2: Variable initialization.

average # variables	% correct outcome	Log-likelihood	Quadratic loss
2 - first stage	**57.52**	0.9332 (0.8900, 0.9765)	0.5514 (0.5206, 0.5821)
2 - second stage	57.19	0.9355 (0.8929, 0.9781)	0.5527 (0.5224, 0.5831)
5 - first stage	56.86	**0.9307 (0.8868, 0.9747)**	**0.5497 (0.5185, 0.5808)**
5 - second stage	56.86	0.9308 (0.8872, 0.9744)	**0.5497 (0.5187, 0.5806)**
10 - first stage	55.88	0.9328 (0.8887, 0.9769)	0.5509 (0.5199, 0.5820)
10 - second stage	56.54	0.9347 (0.8893, 0.9800)	0.5518 (0.5202, 0.5834)
20 - first stage	56.21	0.9340 (0.8892, 0.9787)	0.5523 (0.5209, 0.5837)
20 - second stage	55.88	0.9337 (0.8891, 0.9783)	0.5521 (0.5208, 0.5834)
30 - first stage	56.54	0.9315 (0.8870, 0.9761)	0.5520 (0.5207, 0.5834)
30 - second stage	56.54	0.9315 (0.8865, 0.9765)	0.5519 (0.5203, 0.5835)

Table 7.3: Benchmarks, source: [29]

Model	% correct outcome	Log-likelihood	Quadratic loss
Poisson (ratings)	58.82%	**0.9172 (0.8577, 0.9768)**	**0.5420 (0.4999, 0.5840)**
OLR (ratings)	58.47%	0.9222 (0.8625, 0.9820)	0.5453 (0.5032, 0.5874)
OLR (forecast)	57.52%	0.9332 (0.8760, 0.9904)	0.5506 (0.5106, 0.5905)
Random Forest	57.19%	0.9509 (0.8877, 1.0141)	0.5619 (0.5198, 0.6039)
Bookmaker	**59.80%**	0.9209 (0.8628, 0.9790)	0.5424 (0.5015, 0.5835)
Class frequency	50.65%	1.0292 (0.9910, 1.0674)	0.6188 (0.5917, 0.6459)
Home team	50.65%	3.4092 (3.0217, 3.7968)	0.9869 (0.8747, 1.0991)

Figure 7.3: Ensemble size performance.



Chapter 8

Discussion & Future work

8.1 Discussion

8.1.1 Overall performance & beating the benchmark

What can be seen from figures 7.1 and 7.2 is that the performance of the ensemble model improves as the algorithm advances, which is an indication that the algorithm is working appropriately. Another nice observation is that the ensemble models clearly outperform the single k-nn prediction models.

Although it was not the primary goal of this research, it is still important to consider how good the model did compared to the predictions given by the benchmarks. When taking a look at the the performance of both the bookmakers and the evolutionary ensemble model, it is clear that football match outcomes are hard to predict. Not even 60% of the games can be predicted, which is few given the fact that the home team wins roughly 50% of the time. What can be seen from the results in table 7.1 and figure 7.1, when comparing them to table 7.3, is that the standard model performs worse than the best benchmarks available. The best benchmarks available are the Poisson model and the odds provided by the bookmaker. It has to be stated, however, that these performances are incredibly close and all of the better models are well within each others 90% confidence interval.

The question remains why the model performed worse, albeit not by that much, than the best benchmarks. There are couple of facts that could be the cause of this, the first is the fact that the model tends to overfit on training data, the second might be that the better benchmarks are models that are specialized models for the football domain, while the evolutionary ensemble classifier is not.

8.1.1.1 Model overfitting

During the evolutionary algorithm, when the number of generations increases, the individuals within the population get better at predicting the fitness evaluation set. As can be seen in figure 7.1, ensembles get a performance on this set of around 0.90. Apparently, these are the models that are exceptionally good at predicting that specific set of matches. The performance of the ensembles is worse on the test set. Generally, a score slightly above 0.93 is achieved on this collection of matches. This means that the model does not necessarily find the optimal general football prediction model, but finds a model that is good at predicting the fitness evaluation set. The performance on the test set starts to deteriorate after approximately 20 generations of the evolutionary algorithm. A limitation to this type of modeling could be that it simply needs more data than the benchmark models, so that it is able to find a more general football prediction model.

8.1.1.2 Specialized benchmark models

Another reason for the fact that the evolutionary ensemble model was not able to beat the benchmark might be because of the dataset and the fact that those benchmarks use an inherently different approach, that might be more appropriate for football. The best benchmarks, aside from the unknown bookmaker model, are models that use ratings. Both the Poisson model and the ordered logistic regression model are well founded in the football literature and use a rating based approach. These models did not use the same dataset as the evolutionary ensemble model, but only used ratings based on the performance of the teams on previous games. It is a difficult task to perform better than such a model, that specializes on using past result information, with an unspecialized model. Especially considering the fact that the dataset used by the evolutionary model is made up out of past result information as well. For comparison, the models that used the same dataset as the evolutionary ensemble model, the random forest and the forecast ordered regression model, both performed worse than the evolutionary ensemble model. This last fact indicates that the performance of the evolutionary ensemble model was good, given the current dataset. During the research, the dataset might have been limiting the performance of the evolutionary model, instead of the model itself.

8.1.2 Model comparison

8.1.2.1 First stage vs. second stage

When looking at the log-likelihood, a surprising result is the fact that the performance after the first stage consistently beats the performance of the second stage. The only exception is when the fitness evaluation set is split into a set for the first stage and a set for the second stage. The fact that the first stage beats the second stage consistently can be explained by two reasons. The first reason is that the ensemble created by the first stage had an ensemble size of 250, because that is the size of the final population. Figure 7.3 shows that, at least until an ensemble size of 50, bigger ensembles tend to do better. For the ensembles in the second stage, a size of 40 was chosen. This could be the reason why these ensembles performed worse. This would mean that the initial modeling choice to have an upper bound to the number of individuals in the ensembles of the second stage might be wrong. Apparently, using all the possible individual k-nn predictors in each ensemble is better. The second reason for the worse performance of the second stage might be that this stage simply continues to overfit on the fitness evaluation set and therefore does not give any improvements on the actual test set. The test to split the fitness evaluation set into two separate evaluation sets was done to counter this effect. It does not improve the overall performance of the model, but the second stage seems to perform slightly better than the first stage by using this technique. However, the performances of both stages are hard to compare, since they use different training data.

8.1.2.2 Added features

Changing the fitness evaluation to the prediction of the goal difference did not change the performance of the model. On the contrary, the performance of this model compared to the standard model is almost identical. The conclusion that can be drawn from this result is that the model is robust despite the change in the fitness measure. The fact that those performances are so identical, can considered to be an indication of the stability of the algorithm. The use of the prediction correlation was expected to provide an improved performance over the standard model. This was not the case, the models have a similar performance, with the standard model performing slightly better. The same can be said for the modified mutation approach. Weighted voting does seem to perform better than the original model, as was predicted by the literature [56].

In an attempt to counter the overfitting problem, individuals were tested on random subsets of the fitness evaluation set. The goal here was to create individuals that would specialize less on the total fitness evaluation set. The observed behavior here was that this countered the deterministic crowding algorithm, since individuals in the entire generation would converge to the same point in the solution space.

Changing the amount of variables used by the k-nn classifiers changed the performance of the model. The model has the highest performance when it uses only five variables for the k-nn algorithm. Although using 10 and 20 performed worse than the standard test with 30 variables, so there is no clear relation observed between this amount of variables and the actual performance of the model.

8.2 Future work

This type of evolutionary ensemble model shows promise, although there are a couple of points that could still be further developed. The most important of these points is the fact that the performance of the model seemed limited by the availability and type of data. When a bigger dataset would be used, with most importantly more diversity in the type of variables, this type of prediction model might give better predictions. This type of modeling might just need more data than the rating models that are more commonly applied when predicting football. Applying this kind of prediction model on completely different datasets could also give invaluable information about the quality of the model.

Other work could be done in improving the performance of the second stage of the algorithm. According to the tests, bigger ensembles seem to have better results. The initial assumption, that the size of the ensemble would have a relatively low upper bound, was not proven by the tests. The genotype of the second stage was constructed in such a way that this maximum ensemble size was made possible. Maybe another type of encoding would be more appropriate where each individual is immediately placed in the ensemble, but only the weights of the individuals are optimized. Future work could be directed into different types of encoding for the second stage of the algorithm.

Last but not least, due to the time constraint in this research, there is still a lot of room where tests where not fully conclusive. When more time is available, more effort could be put into more extensive tests to create a complete overview of the performance of this new brand of prediction models.

Chapter 9

Conclusion

In this thesis, a research is described about a novel method for the creation of ensemble prediction models. Ensembles are constructed with the help of an evolutionary algorithm. This new method is motivated by the key feature that both ensemble methods and evolutionary algorithms share, which is that the success of both techniques is highly dependent on the diversity of the individuals that the system is build around. This *evolutionary ensemble model* constructs an ensemble of nearest neighbor models, based on the fuzzy matching system of Sentient's DataDetective. This model is then used to predict the outcomes of football games in the Dutch Eredivisie, although it is not specifically designed to predict football and could be easily applied in any field.

This evolutionary ensemble model consists of two stages. The first stage uses a multi-modal evolutionary algorithm, called deterministic crowding, to find those individual prediction models that work best together. The second stage of the algorithm uses an evolutionary algorithm to optimize the weights of these individuals within the ensemble. This model is tested, along with a couple of added features. The performance of the prediction model is better when fewer variables are used by the nearest neighbor algorithms and when weighted voting is applied.

The model performs only slightly worse than predictions made by bookmakers and other well known football prediction models, but was better than another ensemble method trained with the same dataset. This provides an indication that the evolutionary ensemble model gives robust predictions despite the fact that it is not a specialized football prediction model.

Bibliography

[1] Inderpal Bhandari, Edward Colet, Jennifer Parker, Zachary Pines, Rajiv Pratap, and Krishnakumar Ramanujam. Advanced scout: Data mining and knowledge discovery in nba data. *Data Mining and Knowledge Discovery*, 1:121–125, 1997.

[2] Jürgen Branke, Thomas Kaußler, Christian Schmidt, and Hartmut Schmeck. A multi-population approach to dynamic optimization problems. In *In Adaptive Computing in Design and Manufacturing*, pages 299–308. Springer, 2000.

[3] Leo Breiman. Bagging predictors. In *Machine Learning*, volume 24, pages 123–140, 1996.

[4] Leo Breiman. Random forests. In *Machine Learning*, volume 45, pages 5–23, 2001.

[5] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5 – 20, 2005.

[6] Arjun Chandra and Xin Yao. Divace: Diverse and accurate ensemble learning algorithm. In Zheng Yang, Hujun Yin, and Richard Everson, editors, *Intelligent Data Engineering and Automated Learning IDEAL 2004*, volume 3177 of *Lecture Notes in Computer Science*, pages 619–625, 2004.

[7] Anthony C. Constantinou and Norman E. Fenton. Solving the problem of inadequate scoring rules for assessing probabilistic football forecasts models. *Journal of Quantitative Analysis in Sports*, 8, 2012.

[8] Anthony C. Constantinou, Norman E. Fenton, and Martin Neil. pi-football: A bayesian network model for forecasting association football match outcomes. *Knowledge-Based Systems*, (0), 2012.

[9] Thomas G. Dietterich. Machine learning research: four current directions. *AI Magazine*, 18(4):97–136, 1997.

[10] Mark J. Dixon and Stuart G. Coles. Modelling association football scores and inefficiencies in the football betting market. *Applied Statistics*, 46:265–280, 1997.

[11] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer, oct 2008.

[12] Yoav Freund and Robert Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, volume 904 of *Lecture Notes in Computer Science*, pages 23–37. Springer Berlin / Heidelberg, 1995.

[13] Christian Gagné, Michèle Sebag, Marc Schoenauer, and Marco Tomassini. Ensemble learning for free with evolutionary algorithms? In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1782–1789. ACM, 2007.

[14] Amit P. Ganatra and Yogesh P. Kosta. Comprehensive evolution and evaluation of boosting. *International Journal of Computer Theory and Engineering*, 2(6), December 2010.

[15] John Goddard. Regression models for forecasting goals and match results in association football. *International Journal of Forecasting 21*, 2005.

[16] John Goddard and Ioannis Asimakopoulos. Forecasting football results and the efficiency of fixed-odds betting. *Journal of Forecasting*, 23(1):51–66, 2004.

[17] Anjela Y. Govan, Amy N. Langville, and Carl D. Meyer. Offense-defense approach to ranking team sports. *Journal of Quantitative Analysis in Sports*, 5(1), January 2009.

[18] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10), October 1990.

[19] Georges R. Harik. Finding multimodal solutions using restricted tournament selection. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, San Francisco, CA, 1995. Morgan Kaufmann.

[20] Thomas Haynes and Ip Sen. Crossover operators for evolving a team. In *Proceedings of Genetic Programming 1997: The Second Annual Conference*, pages 162–167. Morgan Kaufmann Publishers, 1997.

[21] Susanne Hoche and Stefan Wrobel. A comparative evaluation of feature set evolution strategies for multirelational boosting. In *Inductive Logic Programming*, volume 2835 of *Lecture Notes in Computer Science*, pages 180–196. Springer Berlin / Heidelberg, 2003.

[22] F. Hoffmann. Boosting a genetic fuzzy classifier. In *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, volume 3, pages 1564 –1569 vol.3, july 2001.

[23] J.H. Holland. Adaptation in natural and artificial systems. *Ann Arbor: University of Michigan press*, 1975.

[24] Ulf Johansson, Rikard König, and Lars Niklasson. Genetically evolved knn ensembles. In Robert Stahlbock, Sven F. Crone, Stefan Lessmann, Ramesh Sharda, and Stefan Voß, editors, *Data Mining*, volume 8 of *Annals of Information Systems*, pages 299–313. Springer US, 2010.

[25] A. Joseph, N.E. Fenton, and M. Neil. Predicting football results using bayesian nets and other machine learning techniques. *Knowledge-Based Systems*, 19(7):544 – 553, 2006.

[26] Dimitris Karlis and Ioannis Ntzoufras. Analysis of sports data by using bivariate poisson models. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 52(3):381–393, 2003.

[27] Albert Hung-Ren Ko, Robert Sabourin, and Alceu de Souza Britto, Jr. Evolving ensemble of classifiers in random subspace. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 1473–1480, New York, NY, USA, 2006. ACM.

[28] Ruud H. Koning. Balance in competition in dutch soccer. *Journal of the Royal Statistical Society. Series D (The Statistician)*, (3), 2000.

[29] Jan Lasek. Football prediction models, 2008.

[30] Christoph Leitner, Achim Zeileis, and Kurt Hornik. Forecasting sports tournaments by ratings of (prob)abilities: A comparison for the euro 2008. *International Journal of Forecasting*, 26(3):471 – 481, 2010.

[31] Jiang-Ping Li, M.E. Balazs, G.T. Parks, and P.J. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 10(3), 2002.

[32] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2, 2002.

[33] Stefan Luckner, Jan Schröder, and Christian Slamka. On the forecast accuracy of sports prediction markets. In *Negotiation, Auctions, and Market Engineering*, volume 2 of *Lecture Notes in Business Information Processing*, pages 227–234. Springer Berlin Heidelberg, 2008.

[34] M. J. Maher. Modelling association football scores. *Statistica Neerlandica*, 36(3):109–118, 1982.

[35] R. Manner and Samir W. Mahfoud. Crowding and preselection revisited. In *Proceedings of the Parallel Problem Solving from Nature Conference*, pages 27–36. North-Holland, 1992.

[36] Byungho Min, Jinhyuck Kim, Chongyoun Choe, Hyeonsang Eom, and R.I. (Bob) McKay. A compound framework for sports results prediction: A football case study. *Knowledge-Based Systems*, 21(7):551 – 562, 2008.

[37] Andrew James Moore. Predicting football results, 2006.

[38] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 798 –803, may 1996.

[39] A. Rotshtein, M. Posner, and A. Rakityanskaya. Football predictions based on a fuzzy model with genetic and neural tuning. *Cybernetics and Systems Analysis*, 41:619–630, 2005.

[40] Mark Rowan. Evolving strategies for predicition of sporting fixtures, April 2007.

[41] Havard Rue and Oyvind Salvesen. Prediction and retrospective analysis of soccer matches in a league. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 49(3):399–418, 2000.

[42] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[43] Benjamin Scheibehenne and Arndt Bröder. Predicting wimbledon 2005 tennis results by mere player name recognition. *International Journal of Forecasting*, 23(3):415 – 426, 2007.

[44] Gulshan Singh and Kalyanmoy Deb, Dr. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 1305–1312, New York, NY, USA, 2006. ACM.

[45] Martin Spann and Bernd Skiera. Sports forecasting: a comparison of the forecast accuracy of prediction markets, betting odds and tipsters. *Journal of Forecasting*, 28(1):55–72, 2009.

[46] M. Srinivas and Lalit M. Patnaik. Genetic algorithms: A survey. volume 27, pages 17–27, 1994.

[47] H.O. Stekler, David Sendor, and Richard Verlander. Issues in sports forecasting. *International Journal of Forecasting*, 26(3):606 – 621, 2010.

[48] J. Sylvester and N.V. Chawla. Evolutionary ensemble creation and thinning. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 5148 –5155, 0-0 2006.

[49] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining.* Pearson Education, Inc., 2006.

[50] Russell Thomason and Terence Soule. Novel ways of improving cooperation and performance in ensemble classifiers. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1708–1715, New York, NY, USA, 2007. ACM.

[51] Andreas Töscher, Michael Jahrer, and Robert M. Bell. The bigchaos solution to the netflix grand prize, September 2009.

[52] A. Tsakonas, G. Dounias, S. Shtovba, and V. Vivdyuk. Soft computing-based result prediction of football games. In *Proceedsings of the First INternational Conference on Inductive Modeling*, volume 3, pages 15–21, 2002.

[53] Chow Kong Wing, Karen Tan, and Lu Yi. The use of profits as opposed to conventional forecast evaluation criteria to determine the quality of economic forecasts, 2007.

[54] Ian H. Witten and Eibe Frank. *Data Mining practical Machine Learning tools and techniques.* Morgan Kaufmann publishers, 2005.

[55] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241 – 259, 1992.

[56] Jakub Zavrel. An empirical re-examination of weighted voting for k-nn. In *Proceedings of the 7th Belgian-Dutch Conference on Machine Learning (BENELEARN-97)*, pages 139–148, 1997.