Universitat Politècnica de Catalunya

Advanced Machine Learning Techniques

CBR Project Report: Technical Manual

Authors:
Pablo Pardo Garcia,
Denis Ergashbaev,
Jeroni Carandell
Saladich,
and Iosu Mendizabal

January 19, 2015

Contents

1 Introduction

For the final AMLT project our group has decided to develop a CBR-based football prediction application, called CBR Bet Assistant. The goal of the system is to predict a likely outcome of given stand-off between two teams based on a history of past performance. Given a team, it produces 3-way output – home team wins ("1" or "H"), away team wins ("2" or "A") or the match results in a draw ("X" or "D") – which could be used by its users to make informed predictions of likely outcome.

In scope of the project we focus on providing a solution for the first division in the Spanish Football league. However, a simple change of the input source (provided it complies with the given data format) would be sufficient to apply the CBR system to other football leagues and countries. A more detailed discussion of the system will be provided in section 4.

The application has been developed with flexibility in mind: although some problem-specific changes to the CBR system were inevitable, we put some extra effort to build a generic representation of the cases as well as CBR stages that could be reused in other applications when need arises. This generic approach undertaken will be discussed in detail in section 3.

Finally, the performance of the system is evaluated in section 5 and the conclusions along with possible improvements are drawn in section 7.

We have taken some time to determine the most appropriate programming language to develop the system and due to various reasons have settled on python 2. The rationale behind this selection as well as the project requirements are discussed in section 6.

2 State of the Art

In this section, we have to distinguish two goals which this project tackles. In the first place, the generic CBR system and secondly the *CBR Bet Assistant*.

The predecessor of CBR systems can be found in the literature around the 1980s with CYRUS (Kolodner (1983)) and IPP (Lebowitz (1983)). More recently Fuchs and Mille (2005) have proposed a modeling of the CBR at the knowledge level and they distinguish between four different type of knowledge models: the conceptual model of the domain describing the concepts use to describe the domain ontology independently of the reasoning; the case model which separates the case in "problem, solution", and track of reasoning; the tasks reasoning models which include a model of specification and other one of tasks decomposition and; reasoning supports model. Some examples of CBR of each of these knowledge models can be found in the literature such as Gaillard et al. (2014) which proposed an ontology independent CBR for Semantic Web using knowledge stored in RDF format.

There is a lot of money at stake in Football predictions and a lot of betting providers are available. This is what makes football prediction such a sought after application yet not necessarily publicly shared. In Joseph et al. (2006) they compare the use of an expert base Naive Bayes versus other systems in predicting the results of a single team in the English league. The advantage of this kind of Naive Bayes is that by constructing a network detailing the important relationships between the factors involved, we get a more interpretable architecture with advantages for example to draw better conclusions the closer the match gets. In these systems for example if a player gets injured, it can be easily introduced provided the system was prepared for it. A clear drawback is that it needs an expert to set up the whole system.

In the Brazilian league, Tavares (2006) uses two different models to compare a Maximum Likelihood Classifier and a Multi-Layer Perceptron more appropriate for soccer predictions using very simple feature vectors based on frequency results weighted by time.

3 Building a Generic CBR System

As mentioned, we have determined to construct a most generic implementation of the CBR system in order to maximize the possibility of being able to reuse its components while at the same time, making the adaptability friendly and intuitive. This intent has resulted in a number of technical decisions that will be outlined in this section.

3.1 Case Representation

The previous report "CBR Component Analisys" has underlined the wide use of the flat feature-based representation of the cases for its simplicity and ease of use (Bergmann). However, we have preferred to employ the object-based representation for the following reasons:

- to be able to display the object hierarchies of our target domain (described in full detail in section 4)
- to structure and group objects according to their concerns

In order to retain abstraction and to be able to reuse the object representation in other CBR projects, we have devised a two-layered object representation. The first layer (located in `core/internal_repr`) represents generic case structures (`CBRclass`, `Case`, and `CaseBase`)). These python classes are able to represent virtually any object hierarchy independent of the domain.

In a way this can be seen as our redefined Object Class specific for CBR Systems.

The second layer, which is more domain specific, builds on top of the "scaffolding" we have created in `core/internal_repr`. The classes are `Match` and `MatchesCaseBase`, located in the module `wrapper.py`. The `Match` class wraps the abstract `Case` structure whereas the `MatchesCaseBase` extends the `CaseBase`.

This way, we have been able to capitalize on the benefits that object-oriented case representation brings about without significantly sacrificing the generality of the system. Our system deals specifically with the `Match` and `MatchesCaseBase` objects which in turn build on top of the generic classes. If need arises to build another CBR application and we choose to employ an object-oriented case representation, we would only need to provide alternative wrapper objects around the `Case` and `CaseBase`. The underlying infrastructure – constructing and managing the object hierarchy of the domain entities – could be used without any modifications.

The approach we have implemented is illustrated in Figure 1, where the generic part is displayed in the top part and the bottom box presents the concrete implementation of the cases and the relevant functions.

3.2 CBR Phases

Similarly, an effort has been made to approach the implementation of the individual CBR phases – retrieve, adapt, revise, and retain – in a generic way. This could best be illustrated with an example. Class `Case`, for instance defines a method `phases.py#retrieve(casebase, case, similarity_function, thr, max_cases)` whose arguments are:

1. a collection of all cases,
2. a case for which similar cases should be found,
3. a similarity function,
4. a threshold value to filter out similar cases, and
5. the maximum number of cases to retrieve

However, the most crucial argument, the similarity function, is not even defined in the generic representation of the case. `Case` only provides a functionality to apply the function, if one is passed, to the collection of cases and filters the results according to the threshold value passed thereafter.

The similarity function is only defined in the `wrapper.py` module, that in addition to defining objects that are particular to our domain – `Match` or `MatchesCaseBase` – also provides implementations of the CBR phases in

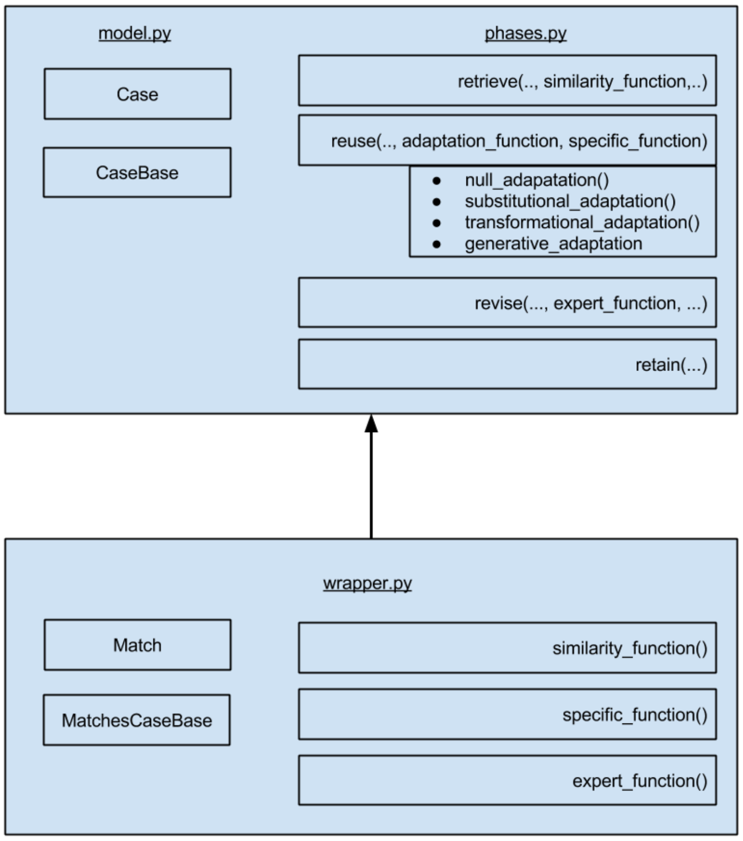
model.py
Case
CaseBase
phases.py
retrieve(.., similarity_function,..)
reuse(.., adaptation_function, specific_function)
null_adapatation()
substitutional_adaptation()
transformational_adaptation()
generative_adaptation
revise(..., expert_function, ...)
retain(...)
wrapper.py
Match
MatchesCaseBase
similarity_function()
specific_function()
expert_function()

Figure 1: Generic CBR System

accordance to the football prediction requirements.

Correspondingly, the other phases of our CBR systems also abstract away from the particular implementation of the relevant functions which are provided by the wrapper module. The generic functions and their respective implementations are listed below.

- `phases.py#reuse(..., adaptation_function, specific_function)`: takes an adaptation function which can be any main adaptation technique described in Wilke (1999) as well as a specific function which is problem related. While we can pick one of the several adaptation functions provided in the generic module `phases.py`, the specific function is defined in the `wrapper.py` module.

- `phases.py#revise(..., expert_function, ...)`: takes an expert function which is again defined in the `wrapper.py` and specific to our scenario.

As can be seen, if we were to apply the CBR application developed to a different domain, we would be able to reuse the generic machinery and the code that commands the CBR cycle without any modification. The only task at hand would be to provide specific implementations of the similarity, adaptation and expert functions as well as devise sensible threshold values. These can thereafter be passed over to the CBR framework and expected to work properly.

4 System Description

Having described the conceptual split line that separates the generic part of the CBR and the part of the system related to the football prediction domain, we would dive deeper into the structure and workflow of the application that has been designed by our group.

4.1 Execution Process

The point of entry is module `main.py`, whose task is to calculate best prediction for a given fixture. Firstly, it generates `Match` objects from the input data (described in Section 4.4 in detail) and stores them in the main memory of the computer. After that, the system performs the 4 CBR phases – retrieve, reuse, revise, and retain – for a given team combination and outputs a prediction as the result of the evaluation of the computations performed. The central entities that are operated on during the run of the algorithm are `Match` and `MatchCaseBase` which are described in the following section.

4.2 Domain Entities: Match and MatchesCaseBase

As previously elaborated, the entities `Match` and `MatchesCaseBase` are concrete necessary extensions of `Case` and `CaseBase` that encapsulate the entities of the problem domain.
`Match` represents a particular match at a given date. It has the following core attributes:

- date – the date (dd/mm/yy) of the game
- home/away – the names of the two teams respectively
- solution – the *known* solution of the stand-off
- multitude of other optional parameters extracted from the data source. These are documented inside the `wrapper.py#Match` source code and are listed in table 1. In essence, these additional features are extra statistics

which are available for a match and could include the following: half-time goal score (home/away), yellow cards, offsides, fouls and different odds and bets from a number of betting providers .

Although we store all this data, in the current version we only use a subset of it as will be discussed in section 4.3.4. It should also be noted, that while the data sources are stored in a file, the cases are kept in memory and are not persisted. At the new start of the system, it will have to retrain (which currently takes negligible amount of time and is not an issue at the moment).

Consecutively, the `MatchesCaseBase` is used group the `Match` entities together and to perform certain operations on them. The `MatchesCaseBase` is able to:

- get the matches a requested team has played as either a home or away team, restricted by time period as specified by the parameter to the method `MatchesCaseBase#get_case_team()`
- generate a chronological record of the home and away plays and their respective dates given teams have participated in (`MatchesCaseBase#get_hist()`)
- fetch a list of matches where the given teams have had a shared opponent (`MatchesCaseBase#get_common_matches()`)

These methods and attributes, though not all of them, are used during the execution of the CBR phases as described below. In the meantime, Figure 2 illustrates some of the most important attributes of these entities.

4.3 CBR Phases

We have followed the classical design of the CBR systems in terms of number of cases without omitting any of them. Following is the description of how each case is implemented in our system.

4.3.1 Retrieve

The retrieve phase applies a similarity function from `wrapper.py#similarity_function` and a threshold value and retrieves the 10 most similar cases from the case base. The similarity function that we use returns a value in the range [0..1] that signifies how similar a case from the case base is to the passed test case, 1 being the most similar value.

In course of development, we have tried a number of similarity measures based of various heuristics, but finally have settled on two.

The first one is the following:

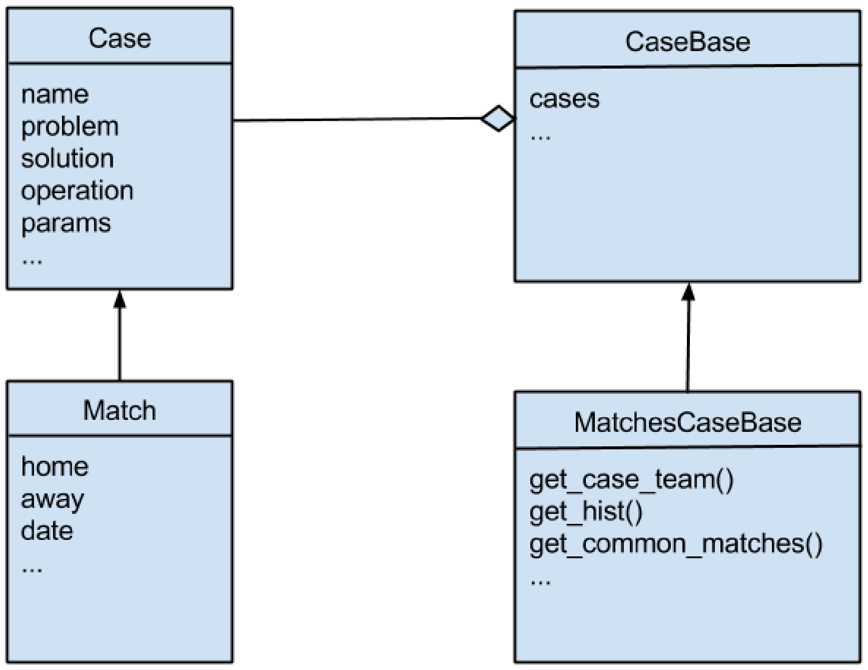

Figure 2: Match and MatchesCaseBase

1. the number of football leagues that separate two cases (matches) is calculated and is then used as a weighting coefficient:

   $w\_years = \frac{1}{1+league\_years\_since\_game}$

   Thus, the longer the time difference between two matches is, the smaller is the weight coefficient

2. if either the home/away correspond exactly between two cases or are merely flipped, the cases are similar where the similarity is calculated in this manner:

   - exact home/away correspondence: $w\_years * 1$
   - flipped home/away correspondence: $w\_years * 0.8$

3. if the matches have different teams, the similarity is deemed to be 0.

The second type uses

- the odds the home team has to win, tie or lose from different providers.
- the number of leagues from the new match like before.

4.3.2 Reuse

In the reuse phase we resort to the substitutional adaptation strategy, which in turn employs `wrapper.py#specific_function()` to calculate the result. We used a rather straightforward approach to predict a possible outcome of the given match, which could be illustrated in the sequence of steps:

1. compare a given match with similar matches returned by the retrieve stage
2. for each comparison, calculate the probability of one of the tree possible outcomes – home team wins ("H"), away team wins ("A") or the match results in a draw ("D"):
   - if home team of the similar wins and the input match's home team corresponds to the similar match's home team, augment the probability of win. Otherwise, augment the probability of a loss.
   - the probability is set by the formula: $probability_{win|lose} += 1 * similarity\_value_{similar\_match}$
   - if there was a draw, augment the probability of draw for the input match
   - if no similar cases have been produced by the retrieve phase, produce arbitrary outcome.
3. finaly a voting is done to choose the final result among the three options.

Since the CBR system is general enough to adapt to other problems from different domains, we have added the integration of general adaptation techniques as mentioned in section 3.2. These are generic types of the main adaptation classifications in (Wilke, 1999) and can be seen in 3. In particular we have developed generic functions for *null_adaptation*, *substitutional_adaptation* and a *structural adpatation* representing the transformational adaptations and on the other hand a generic *generative_adaptation* function which represents the others. Each of these functions receive as input a specific domain function which makes the code easily adaptable to any domain.

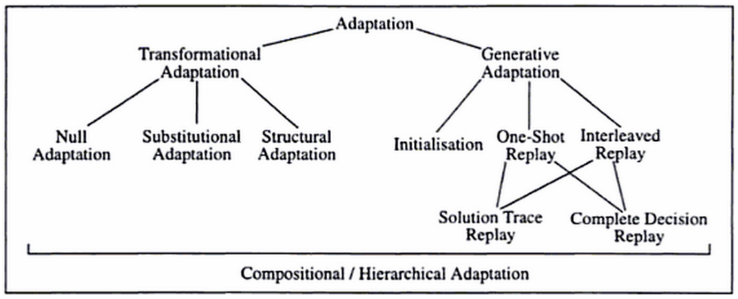

Figure 3: Reuse classification tree.

4.3.3 Revise

Revise employs an expert function to determine a confidence of the solution. In our system, the expert function merely compares the actual result (because we know it from the test set) with the predicted result and returns 0 or 1 correspondingly. This happens as part of the CBR cycle without an interruption and used later to evaluate the accuracy of the application.

It can be envisioned, that under the real-time operation of the system, we would need to delay the revise phase until a real result is known to be able to proceed to the next phase.

4.3.4 Retain

In conclusion, the retain phase is executed. At this stage we evaluate the confidence given by the revise phase. If the confidence is above a confidence threshold *and* no similar cases above a certain similarity threshold exist in our case base, we persist the case and the solution in our case base.

4.4 Datasets

The dataset used for the experiments can be found in `http://www.football-data.co.uk/spainm.php` and consists of 20 csv files consisting each one of all the matches in a season. The first season we have is 1995-1996 and the last one 2013-2014, as well as half of the 2014-2015 season.

Each row of the csv represents a match and for each match we have information about the number of goals scored by individual teams, the betting odds given by different betting providers. The parameters and its descriptions are listed in the Table 1 in the Appendix A. The oldest seasons do not have all the features such a betting odds since they where not available back then.

5 System Performance

We have evaluated the system with the data from the last season, from 2013-14, and having as a Case Base the seasons from the Spanish first division from 2000 to 2012.

5.1 Model Selection

In practical terms, evaluating CBR systems is a difficult task. This is because adaptive systems have an inherent property which makes system comparisons difficult. Weibelzahl and Lauer (2001) propose a theoretical approach to said evaluation in terms of user feedback. Basically the idea is that a CBR system is good if its performance increases over time or at least it does not decrease.

In order to find the best parameters we made a grid search on the test set going through all the different possibilities. The system does not require a distinct learning phase, as it fine-tunes the parameters in an on-line fashion. The decision of retaining or not a particular case (given a definite expert evaluation), as well as what weights to assign to different features undergo continuous evaluation.

In the Figure 4 we can observe the learning curve of the best performing models for each one of the two similarity functions. The accuracy using both similarities is close to 45%, we can see in the learning curve that the accuracy behaves in a very similar manner in both models converging on time to the final accuracy.

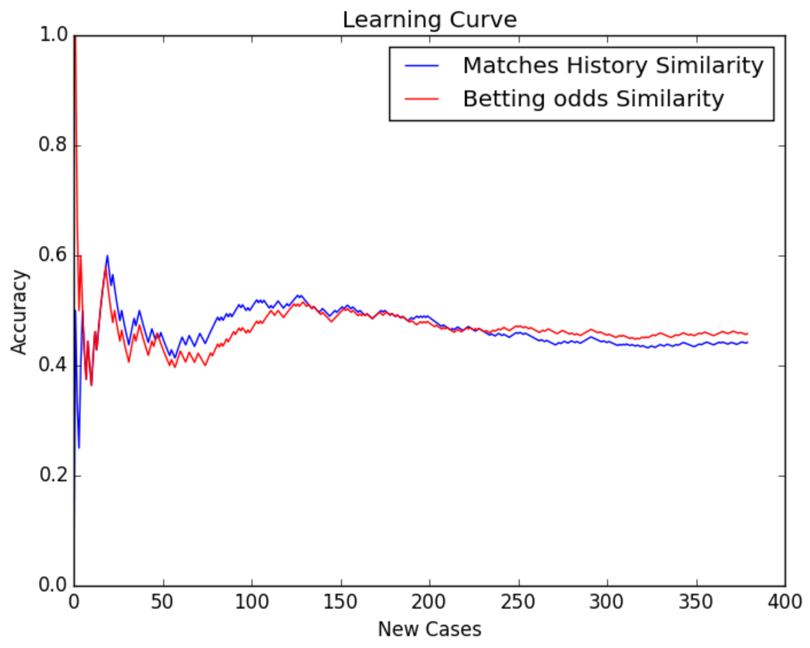

Figure 4: Learning Curves using two different similarity functions.

5.2 Evaluation

There are two modes in which the evaluation phase of the CBR can be performed:

1. Providing an overall prediction accuracy measurement for the test dataset. This is done by looping through all the entries in the test file and making a prediction for each of the matches.

2. Performing a prediction for a particular team fixture that a system user is interested in.

Internally, the system assigns different confidence values for each of the tree possible outcomes, 1/X/2. However, only the outcome with the highest probability is reported.

6 Technical Discussion

While the bundled “User Manual” document provides a user-centric perspective of the CBR Bet Assistant, this section gives an overview of the reasons behind technology decisions taken, the tools used to streamline the development of the project and the documentation of the code.

We pondering on the optimal choice in terms of the programming language to be used. Although all the team members knows both Java and Python, we have decided to develop the CBR system in python for a number of reasons:

- throughout the MAI course we have been been able to polish our skills in Python on several projects
- given different backgrounds of the team members – B.Sc. degrees in mathematics, computer science, and business computing – Python seems to be the best common denominator allowing each team member to contribute with their knowledge unhampered by the programming environment
- we had accumulated experiences with several Python libraries which help produce clear and readable code.

In the course of development we have strictly adhered to the project requirements of not using any third-party Python libraries in the area of case-based reasoning. All the entities and the operations are hand-written and fine-tuned and rely exclusively on the concepts taught in the lecture, research made and previous knowledge of the team members.

6.1 Documentation

Extra care has been given to ensure a well-documented project. Each class and their public methods are properly documented (using a python-style documentation conventions are detected by the PyCharm development studio for type inference and suggestions).

6.2 Version Control

We have used git version control system to manage the manage the code and GitHub as a hosting provider. The code of the project as well as the input data can be found under `https://github.com/Jeronics/CBR-system`. Together with the project documentation, the use of version control system has helped us to streamline the project development under a tight schedule and coordinate the input of the team members.

7 Conclusions and Furture Work

In this paper, we explain our implementation and design of a generic CBR system. In order to validate the latter we have also developed a domain specific application called *CBR Bet Assistant*, which consists of a football result predictor.

Regarding the *CBR Bet Assistant* we were quite impressed with the results considering the difficulty of the problem and the lack of detailed information. However it is still inferior in terms of performance to state-of-the-art solutions which use implementations such as Naive Bayes or neural network. This leads us to believe that improving is possible and more similarity functions should be designed and tested, possibly considering all the features together in the same heuristic.

As future work we propose an alternative implementation which would aim to recommend betting investments in a way to maximize the economical benefits of betting as opposed to the current result classification system. Thus we could provide an application that would fully enhance the use of a CBR system since we could learn to adapt the system to the user's betting tendencies (higher or lower risk).

The generic CBR system has proven to be easily adaptable as we have seen while coding on our specific domain (football score prediction). Furthermore, the object base implementation allows object hierarchies structure representation as well as flat featured ones.

Finally, we conclude that this generic CBR is a template for any CBR system and therefore facilitates the implementation process in terms of both programming and planning time.

References

R. Bergmann. Introduction to case-based reasoning. URL `www.dfki.uni-kl.de/~aabecker/Mosbach/Bergmann-CBR-Survey.ps`. [Online; accessed 2014-12-27].

B. Fuchs and A. Mille. Une modélisation au niveau connaissance du raisonnement à partir de cas, June 2005. URL `http://liris.cnrs.fr/publis/?id=1884`. INGÉNIERIE DES CONNAISSANCES Coordonné par R. Teulier, J. Charlet, P. Tchounikine éditeur L'Harmattan.

E. Gaillard, L. Infante-Blanco, J. Lieber, and E. Nauer. Tuuurbine: A generic cbr engine over rdfs. In L. Lamontagne and E. Plaza, editors, *Case-Based Reasoning Research and Development*, volume 8765 of *Lecture Notes in Computer Science*, pages 140–154. Springer International Publishing, 2014. ISBN 978-3-319-11208-4. doi: 10.1007/978-3-319-11209-1_11. URL `http://dx.doi.org/10.1007/978-3-319-11209-1_11`.

A. Joseph, N. E. Fenton, and M. Neil. Predicting football results using bayesian nets and other machine learning techniques. *Know.-Based Syst.*, 19(7):544–553, Nov. 2006. ISSN 0950-7051. doi: 10.1016/j.knosys.2006.04.011. URL `http://dx.doi.org/10.1016/j.knosys.2006.04.011`.

J. L. Kolodner. Reconstructive memory: A computer model. *Cognitive Science*, 7(4):281–328, 1983. URL `http://dblp.uni-trier.de/db/journals/cogsci/cogsci7.html#Kolodner83a`.

M. Lebowitz. Memory-based parsing. *Artif. Intell.*, 21(4):363–404, Nov. 1983. ISSN 0004-3702. doi: 10.1016/S0004-3702(83)80019-8. URL `http://dx.doi.org/10.1016/S0004-3702(83)80019-8`.

A. T. Tavares. Predicting football results using bayesian nets and other machine learning techniques. 2006. URL `http://homepages.cae.wisc.edu/~ece539/fall13/project/TrindadeTavares_rpt.pdf`.

S. Weibelzahl and C. U. Lauer. Framework for the evaluation of adaptive cbr-systems. In H.-P. Schnurr, S. Staab, R. Studer, G. Stumme, and Y. Sure, editors, *Professionelles Wissensmanagement - Erfahrungen und Visionen*, pages 254–263. Shaker, Aachen, 2001. URL `http://www.easy-hub.org/stephan/weibelzahl-gwcbr01.pdf`.

W. Wilke. *Knowledge management for intelligent sales support in electronic commerce.*, volume 213 of *DISKI*. Infix, 1999. ISBN 978-3-89601-213-5.

A Data Parameters

Table 1: Match information

Features	**Description**
Date	Date of the match
Div	Division of the match
HomeTeam	Home Team
AwayTeam	Away Team
FTHG	Full Time Home Goals
FTAG	Full Time Away Goals
FTR	Full Time Result
HTHG	Half Time Home Goals
HTAG	Half Time Away Goals
HTR	Half Time Result
Attendance	Crowd Attendance
Referee	Match Referee
HS	Home Team Shots
AS	Away Team Shots
HST	Home Team Shots on Target
AST	Away Team Shots on Target
HHW	Home Team Hit Woodwork
AHW	Away Team Hit Woodwork
HC	Home Team Corners
AC	Away Team Corners
HF	Home Team Fouls Committed
AF	Away Team Fouls Committed
HO	Home Team Offsides
AO	Away Team Offsides
HY	Home Team Yellow Cards
AY	Away Team Yellow Cards
HR	Home Team Red Cards
AR	Away Team Red Cards
HBP	Home Team Bookings Points (10 = yellow, 25 = red)
ABP	Away Team Bookings Points (10 = yellow, 25 = red)
B365H	Bet365 home win odds
B365D	Bet365 draw odds
B365A	Bet365 away win odds
BSH	Blue Square home win odds
BSD	Blue Square draw odds
BSA	Blue Square away win odds
BWH	Bet&Win home win odds

Continued on next page

Table 1 – continued from previous page

Features	Description
BWD	Bet&Win draw odds
BWA	Bet&Win away win odds
GBH	Gamebookers home win odds
GBD	Gamebookers draw odds
GBA	Gamebookers away win odds
IWH	Interwetten home win odds
IWD	Interwetten draw odds
IWA	Interwetten away win odds
LBH	Ladbrokes home win odds
LBD	Ladbrokes draw odds
LBA	Ladbrokes away win odds
PSH	Pinnacle Sports home win odds
PSD	Pinnacle Sports draw odds
PSA	Pinnacle Sports away win odds
SOH	Sporting Odds home win odds
SOD	Sporting Odds draw odds
SOA	Sporting Odds away win odds
SBH	Sportingbet home win odds
SBD	Sportingbet draw odds
SBA	Sportingbet away win odds
SJH	Stan James home win odds
SJD	Stan James draw odds
SJA	Stan James away win odds
SYH	Stanleybet home win odds
SYD	Stanleybet draw odds
SYA	Stanleybet away win odds
VCH	VC Bet home win odds
VCD	VC Bet draw odds
VCA	VC Bet away win odds
WHH	William Hill home win odds
WHD	William Hill draw odds
WHA	William Hill away win odds
Bb1X2	Number of BetBrain bookmakers used to calculate match odds averages and maximums
BbMxH	Betbrain maximum home win odds
BbAvH	Betbrain average home win odds
BbMxD	Betbrain maximum draw odds
BbAvD	Betbrain average draw win odds
BbMxA	Betbrain maximum away win odds
BbAvA	Betbrain average away win odds

Continued on next page

Table 1 – continued from previous page

Features	Description
BbOU	Number of BetBrain bookmakers used to calculate over/under 2.5 goals (total goals) averages and maximums
BbMx>2.5	Betbrain maximum over 2.5 goals
BbAv>2.5	Betbrain average over 2.5 goals
BbMx<2.5	Betbrain maximum under 2.5 goals
BbAv<2.5	Betbrain average under 2.5 goals
GB>2.5	Gamebookers over 2.5 goals
GB<2.5	Gamebookers under 2.5 goals
B365>2.5	Bet365 over 2.5 goals
B365<2.5	Bet365 under 2.5 goals
BbAH	Number of BetBrain bookmakers used to Asian handicap averages and maximums
BbAHh	Betbrain size of handicap (home team)
BbMxAHH	Betbrain maximum Asian handicap home team odds
BbAvAHH	Betbrain average Asian handicap home team odds
BbMxAHA	Betbrain maximum Asian handicap away team odds
BbAvAHA	Betbrain average Asian handicap away team odds
GBAHH	Gamebookers Asian handicap home team odds
GBAHA	Gamebookers Asian handicap away team odds
GBAH	Gamebookers size of handicap (home team)
LBAHH	Ladbrokes Asian handicap home team odds
LBAHA	Ladbrokes Asian handicap away team odds
LBAH	Ladbrokes size of handicap (home team)
B365AHH	Bet365 Asian handicap home team odds
B365AHA	Bet365 Asian handicap away team odds
B365AH	Bet365 size of handicap (home team)