EAS 499 Final Thesis

Ali Kozlu

Dec 2018

The preface pretty much says it all.

Second paragraph of abstract starts here.

# Introduction

In recent years, there has been a noticeable rise in sports betting and gambling promotion during sporting events. The two main drivers of growth are the rise of online sports betting platforms and the introduction of “in play betting”, where consumers can place a bet on a range of possible outcomes both after the game has commenced and, as they occur during a game. The rise of the online sports betting volume has stimulated a dramatic increase in sports-related financial activity. Large monetary amounts involved in betting became a primary motivator for research being conducted on prediction models and applications of statistical analysis to the domain of sport. As one of the most popular sports globally, Tennis has become an ideal candidate for modelling and predicting outcome of professional matches. In addition, professional singles tennis is an attractive sport to model algorithmically for a number of reasons. In tennis, each match is played between just two players, as opposed to the multitude of players involved in a team-based game such as football, rugby or basketball. Because of this there is no need to analyse the offensive and defensive strengths of possible combinations of starting line ups. Also, the large amount of historical data freely available makes the sport of Tennis an ideal candidate for Machine learning (ML) algorithms, which have shown promising results in the domains of classification and prediction. The goal of the present thesis is to investigate the applicability of machine learning methods to the prediction of professional tennis matches. In doing so, we explore appropriate means of feature engineering from data sources, model evaluation, and specific challenges of predicting tennis results.

# Background

## Tennis Datasets

As mentioned in the introduction, there is a large amount of historical tennis data freely available. Tennis websites such as [atpworldtour.com](www.atpworldtour.com) and [tennis.data.co.uk](www.tennis-data.co.uk/alldata.php) provide most comprehensive access to information about players, the outcomes of matches and statistics of certain matches. Other important sources that include historical datasets (as CSV) of ATP Tennis Rankings, Results, and Stats are provided by [Jeff Sackman](https://github.com/JeffSackmann/tennis_atp) and [Daniel Korzekwa](https://github.com/danielkorzekwa/atpworldtour-api). A more complex dataset with a longer historical timespan and advanced metrics is provided by the [OnCourt System](www.oncourt.info). The present thesis uses the OnCourt Dataset, which includes statistics of 17625 ATP tennis matches played between 2004-2018 and has results for over 500 thousand ATP matches played since the 1990’s. OnCourt dataset also contains further information such as tournament seeding, tournament round, player age and length of match. OnCourt Dataset, which is in MDB format, is converted into a Sqlite3 database using [Mdb Tools](https://github.com/brianb/mdbtools). A Pyton script then extracts different information from Sqlite3 Database (tournaments, player names, statistics) and converts the data into Pandas Dataframe.[[1]](#footnote-30) The full list of statistics found in OnCourt dataset is omitted, as our model uses none of this additional information. The relevant statistics provided for each match can be seen in the figure below.

Table 1 Per Match Statistics found in OnCourt Dataset

|  |  |
| --- | --- |
| OnCourt Match Statistics |  |
| First Serve percentage | FS |
| Percentage of points won on first serve | WSP\_1 |
| Percentage of points won on second serve | WSP\_2 |
| Percentage of receiving points won | WRP |
| Double Faults | DF |
| Aces | ACES |
| Total Points Won | TPW |
| Break Points Won | BP |
| Total Break Points | TBP |
| Net Approaches | NA |
| Total Net Approaches | TNA |

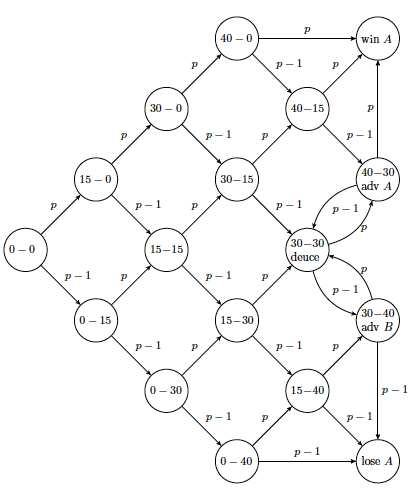
## Tennis Betting

There are two main categories of tennis betting: pre-play and in-play. The former is considered as a traditional form of betting, in which a bookmaker offers odds and accepting punters bet directly against the bookmakers. The second category can be played in betting exchanges such as [Bet Fair](https://www.betfair.com/exchange/plus/), in which customers instead offer odds and place bets against each other. A unique option brought by exchanges, called lay betting, allows punters to oppose a selection, i.e. to bet against something happening. In this case, punters can play the role of a traditional bookmaker but offering odds to sell a bet instead of the usual odds to back a bet. In both categories it is possible to bet on a variety of factors, such as such as the winner of the match, the number total sets and the over/under of total number of games. Traditionally, research has focused on bets placed on the overall outcome prior to the match starting as it allows a more comprehensive evaluation of the performance of the generated model against the market. However, experts claim that over 80% of the overall money wagered on tennis matches is bet in-play, i.e., during the course of the match. Thus, instead of only focusing on pre-play, the present thesis attempts to use the model for both pre-play and in-play betting to see the true value of our model. Webb (2011). (Also <https://www.dailymail.co.uk/sport/tennis/article-3405544/Tennis-gambling-market-second-football-bookmakers-boom-online-play-betting.html>).

## Review of Previous Work

The earliest attempts to the statistical analysis of tennis matches can be traced back to early 2000’s. S. Clarke & Dyte (2000) and Boulier & Stekler (1999) used a single feature, namely ATP computer tennis rankings, to predict a player’s chance of winning. The goal was to see whether the position of a tennis player in world rankings was related to his/her performance in an upcoming tournament. These early attempts fit a logistic regression model to the ATP ratings, to estimate the probability of winning as a function of the difference in rating points. Del Corral & Prieto-Rodriguez (2010) attempt to assess the degree to which the difference in ranking points is good indicators of the outcome of Grand Slam matches. More recently, Spanias & Knottenbelt (2013) claimed that the ATP ranking system is inaccurate at ranking players with rankings greater than 32, showing that ATP rankings perform much more poorly in this subset of matches with a success rate as low as 55%. Another problem is that there is only one ranking for the four surfaces, thus ranking system does not capture player strengths on different surfaces. More recent state-of-the-art tennis prediction models can be seen as a hierarchical/point model using Markov chains. These models take advantage of the fact that scoring system in tennis has a hierarchical structure, with a match being composed of sets, which are composed of games, which in turn are composed of individual points. Thus one can construct a hierarchical Markov model to estimate the match winning probabilities of both players by using *only the probabilities of the two players winning points while serving*. Using this singular statistic, one can deduce the probability of a player winning a single point, then a game, then a set, and finally the match, since the Markov chain will reflect the stochastic progression of the score in a game, as in 1. Hertzmann & Zorin (2001) Newton & Keller (2005) James (2008) used Markov chains to generate equivalent hierarchical expressions using the assumption that points are independent and identically distributed. Klaassen & Magnus (2003) later show that winning the previous point can have a positive influence on winning the current point, but still argue that the model is good approximatio since tnhe deviations from i.i.d. are small. Arguing that small deviations does not unjustify the i.i.d. assumption in forecasting, Barnett & Clarke (2005), Spanias & Knottenbelt (2013) and Newton & Keller (2005) focus in detail on the estimation of the respective probability of each player winning a point while serving. In order to calculate this probability, Barnett & Clarke (2005) attempt to mathematically combine the serving ability of the serving player against the returning ability of the returning player. In order to come up with these two numbers, Barnett & Clarke (2005)’s approach is to can look at historical service statistics of both players and average each player’s statistics. As explained in Spanias & Knottenbelt (2013), Barnett’s approach computes two statistics for each player reflecting their serving and receiving strengths against an average opponent respectively, where are computed as:[[2]](#footnote-36)

In which,



Although mathematically sound, one can argue that this approach is far from perfect empirically. Even if is a good approximation; Barnett’s algorithm suffers from bias because players face different opponents of varying skill levels. Knottenbelt’s Common Opponent Model attempts to solve this flaw with Barnett’s algorithm. He adopts the way the serve-winning probabilities of players are calculated before being supplied to the Barnett formulas by only looking at the players’ performance against *common opponents*. The crucial point is that this version provides a fair basis of comparison between players by analysing match statistics for opponents that both players have encountered in the past. Knottenbelt claims that his common opponent hierarchical Markov model yields 3.8% ROI against the best odds offered by bookmakers for a data set of 2178 diverse tennis matches, but admits that the sample size is big enough to represent the average performance of the model.

Perhaps more importantly, the hierarchical Markov models do not take other important match statistics such as number of aces, double faults, total points won and second serve winning percentage, into account. Hence it is unrealistic to expect that these models can successfully capture important factors such as playing styles of players, fatigue and individual player strengths. Considering the availability of an immense amount of historical tennis data, recently several master theses have proposed machine learning models as an alternative to stochastic models described above. Somboonphokkaphan, Phimoltares, & Lursinsap (2009) train neural networks with basic features features, including previous head-to-head match outcomes and first serve percentage etc. The authors claim that the model has 75% accuracy for predicting matches in the 2007 and 2008 Grand Slam tournaments. Chen, Tian, & Zhong (2017) try support vector classification models (SVC) with linear, RBF and polynomial kernels, but the results of the thesis are inconclusive. A more comprehensive paper by Sipko (2015), whose thesis advisor was William Knottenbelt, uses the OnCourt dataset to train a Logistic Regression with interaction features and a Feed-Forward Neural Network Model.Sipko claims the neural network model generated a return on investment of 4.4% when betting on 6315 ATP matches in 2013-2014..[[3]](#footnote-38) Knottenbelt and Sipko use the above mentioned common opponent model to derive not only the proability of winning a point, but all the possible variables representing the qualities of two players, mentioned in Figure (Oncorut dataset). Sipko’s paper is significant not only because of the models it introduces, but also because the novel feature engineering methods it uses to extract tennis match features from raw historical data, characterizing the qualities of two players in the most representative way. The following thesis closely follows the novel feature engineering done by Sipko, which is explained in the next chapter.

## Including plots

You can also embed plots. For example, here is a way to use the base **R** graphics package to produce a plot using the built-in pressure data set:



Note that the echo=FALSE parameter was added to the code chunk to prevent printing of the **R** code that generated the plot. There are plenty of other ways to add chunk options. More information is available at <http://yihui.name/knitr/options/>.

Another useful chunk option is the setting of cache=TRUE as you see here. If document rendering becomes time consuming due to long computations or plots that are expensive to generate you can use knitr caching to improve performance. Later in this file, you’ll see a way to reference plots created in **R** or external figures.

## Loading and exploring data

Included in this template is a file called flights.csv. This file includes a subset of the larger dataset of information about all flights that departed from Seattle and Portland in 2014. More information about this dataset and its **R** package is available at <http://github.com/ismayc/pnwflights14>. This subset includes only Portland flights and only rows that were complete with no missing values. Merges were also done with the airports and airlines data sets in the pnwflights14 package to get more descriptive airport and airline names.

We can load in this data set using the following command:

flights <- read.csv("data/flights.csv")

The data is now stored in the data frame called flights in **R**. To get a better feel for the variables included in this dataset we can use a variety of functions. Here we can see the dimensions (rows by columns) and also the names of the columns.

dim(flights)

[1] 52808 16

names(flights)

[1] "month" "day" "dep\_time" "dep\_delay"   
 [5] "arr\_time" "arr\_delay" "carrier" "tailnum"   
 [9] "flight" "dest" "air\_time" "distance"   
[13] "hour" "minute" "carrier\_name" "dest\_name"

Another good idea is to take a look at the dataset in table form. With this dataset having more than 50,000 rows, we won’t explicitly show the results of the command here. I recommend you enter the command into the Console ***after*** you have run the **R** chunks above to load the data into **R**.

View(flights)

While not required, it is highly recommended you use the dplyr package to manipulate and summarize your data set as needed. It uses a syntax that is easy to understand using chaining operations. Below I’ve created a few examples of using dplyr to get information about the Portland flights in 2014. You will also see the use of the ggplot2 package, which produces beautiful, high-quality academic visuals.

We begin by checking to ensure that needed packages are installed and then we load them into our current working environment:

# List of packages required for this analysis  
pkg <- c("dplyr", "ggplot2", "knitr", "bookdown", "devtools")  
# Check if packages are not installed and assign the  
# names of the packages not installed to the variable new.pkg  
new.pkg <- pkg[!(pkg %in% installed.packages())]  
# If there are any packages in the list that aren't installed,  
# install them  
if (length(new.pkg))  
 install.packages(new.pkg, repos = "http://cran.rstudio.com")  
# Load packages (thesisdowndss will load all of the packages as well)  
library(thesisdowndss)

The example we show here does the following:

## Additional resources

* *Markdown* Cheatsheet - <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>
* *R Markdown* Reference Guide - <https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf>
* dplyr Documentation - <http://dplyr.tidyverse.org/>
* ggplot2 Documentation - <http://ggplot2.tidyverse.org/>

# Feature Engineering

## Feature Extraction

Any tennis prediction model must be able to represent the skill difference between both players. Since our dataset includes two x observations (of the same variable) uniformly sampled in the real numberrange [0,1], taking the difference or ratio for two values of same variable becomes natural choices for feature engineering. For example, if we wanted to compute the ACES variable for an upcoming match, we simply do: . This approach also cuts the dimension of our feature space in half, which is important in machine learning because the amount of data required to provide a reliable analysis grows exponentially with the dimensionality of features. The so-called *large p, small n* problem, where p is the number of features and n is the number of samples, tend to be prone to overfitting. An overfitted model can mistake small fluctuations for important variance which can lead to classification errors. One can argue that using two observations of the same statistic can preserve more information and give our model a better chance to find certain patterns, but in practice the difference in statistics of two player’s has enough predictive performance.[[4]](#footnote-52) So we will be using the differences in variables as our match features. Any effective tennis prediction model must also consider the most representative set of features to capture all possible qualities of both players.Constructing or selecting features from tennis statistics requires a deep understanding of the sport and the true meaning behind the numbers. Thus, a significant challenge for any tennis prediction model is the transformation match statistics into the most representative match features. In the present work, this transformation is done in two major phases, Feature Selection and Feature Construction.

Table 2 Difference Transformation for Generating Final Features

|  |  |  |
| --- | --- | --- |
| Avg. Return % for P1 | Avg. Return % for P2 | Final Feature |
| 0.58961 | 0.14544 | 0.44417 |
| 0.66203 | 0.98726 | -0.32523 |
| 0.89746 | 0.58317 | 0.31429 |
| 0.06489 | 0.11745 | -0.05256 |
| 0.44753 | 0.34509 | 0.10244 |

## Feature Selection

### Computing Advanced Statistical Metrics

The most innovative aspect about Sipko and Knottenbelt’s work is how they apply their knowledge of the tennis domain to generate advanced metrics using player’s performance statistics found in OnCourt Dataset. Three metrics that Sipko and Knottenbelt introduce that are part of our feature set are:[[5]](#footnote-55)

#### Overal Winning on Serve Percentage

The idea behind this metric is to combine the winning percentage on first and second serve found in OnCourt dataset to come up with a single metric that quantifies a player’s *strength when serving*. WSP is calculated in the following way for player i:

*= Winning % on first Serve*  
 *= Winning % on second Serve*  
 *= First Serve Percentage.*

#### Completeness

Similar to WSP, Completeness combines a player’s serve and return percentages in order to combine their offensive and defensive game. For player i:

*= Overal Winning on Serve Percentage*  
 *= Overal Winning on Receiving Points percentage*

#### Advantage on Serve

This is the only metric in our dataset that combines information from both players. The idea of Knottenbelt and Sipko is to grasp the difference between the serving player’s serve strength and the returning player’s return strength, thereby quantifying how much advantage serving player has over the returning player. The authors name this feature SERVEADV (player’s advantage on serve). The feature is computed in the following way:

#### Additional Features

**Head to Head**  
This metric takes the outcomes of matches between two players into account. For any two players in a given match, we look at the past matches between two players and calculate the number of matches both player has won. Afterwards Head-to-Head feature is calculated as follows for players i and j:

**Aces and Percentage of Total Points Won**  
These metrics quantify the total number of Aces and Percentage of Total Points Won by a player in a match. Since these values are dependent on the number of games in a match, these features will be normalized later.

**Percentage of Breaking Points**  
This feature quantifies the percentage of break points won in that match for both players. If there were no breaking points in the match, that match is dropped from the dataset. Also breaking point percentages that are not in range [0,1] are dropped to reduce noise in the data. Adding this feature is a controversial decision, because they are few observations (breaking points) in a match, which inevitably result in extreme probabilities. However, this issue is partly alliviated by how we construct features, which is explained in the next section.  
**Additional Information**  
For every match, the total number of games, the total number of sets, the year and the court type of the match are also added to our data frame, as these information are needed for construction our final features. Computing number of games and sets also give us flexibility for further work, as we are also interested in predicting number of games and sets. A Python script is responsible extracting advanced metrics, head to head statistics and additional information from the initial OnCourt per match statistics. The resulting dataframe is converted into a Sqlite3 Database for further use. The first thirty rows of our initial dataset and the resulting database can be seen below.[[6]](#footnote-60)

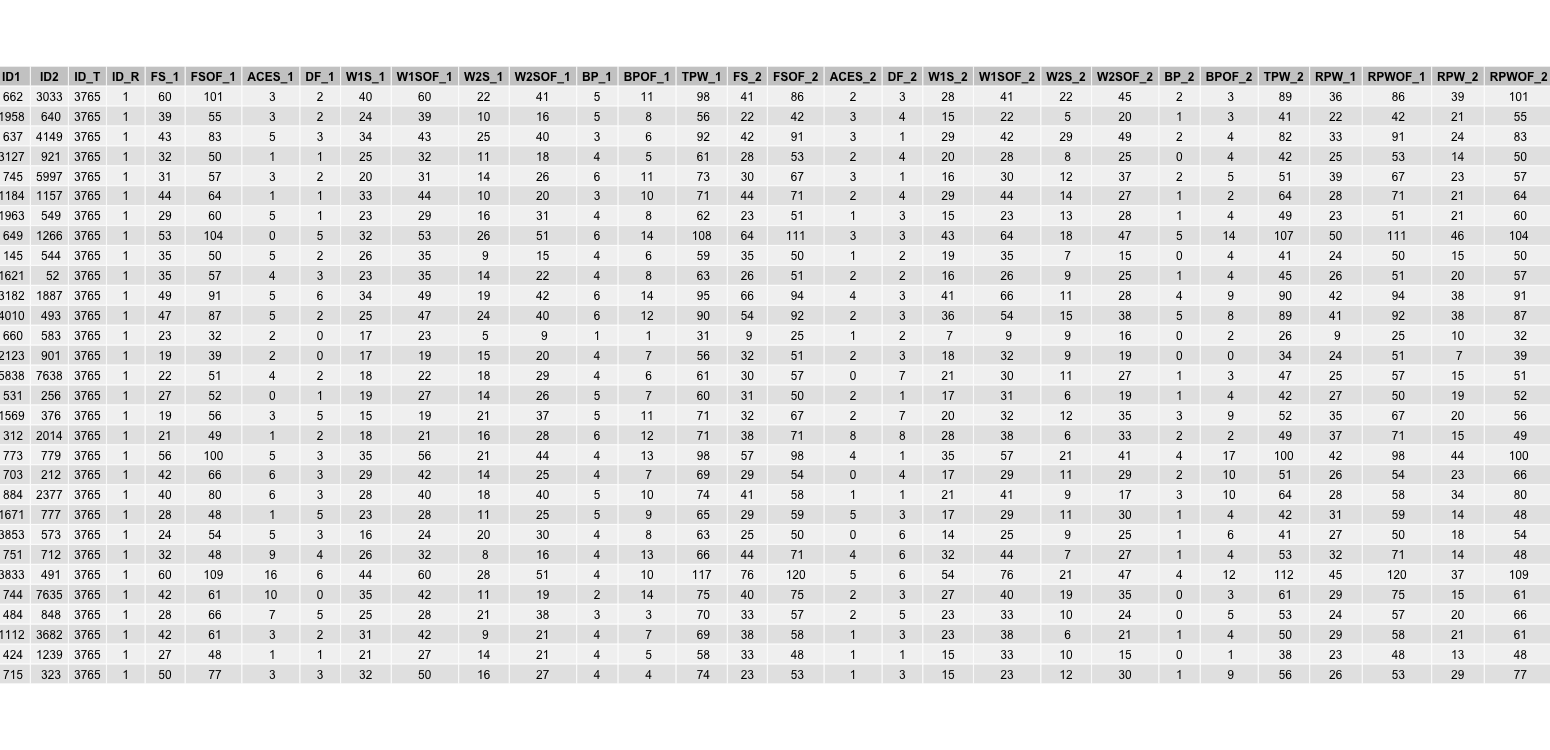
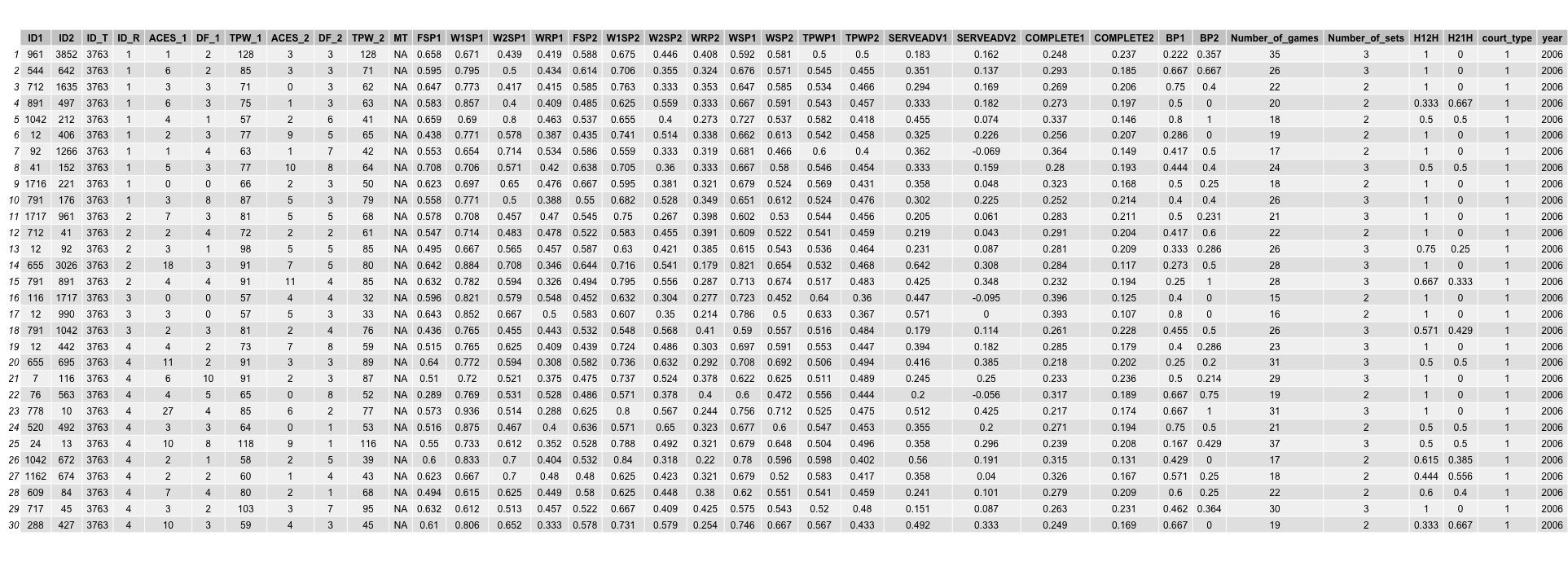
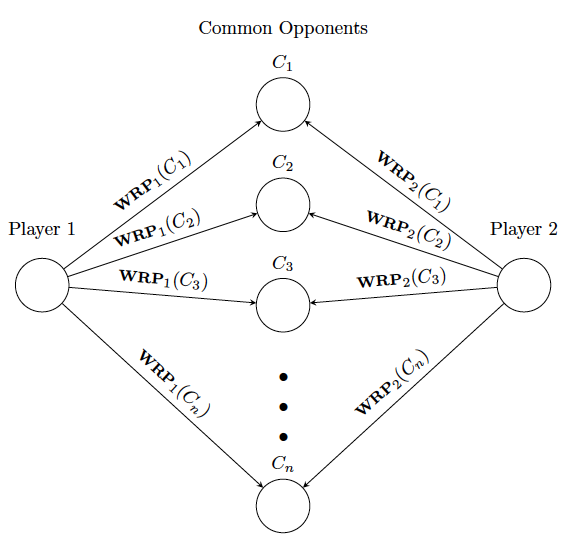
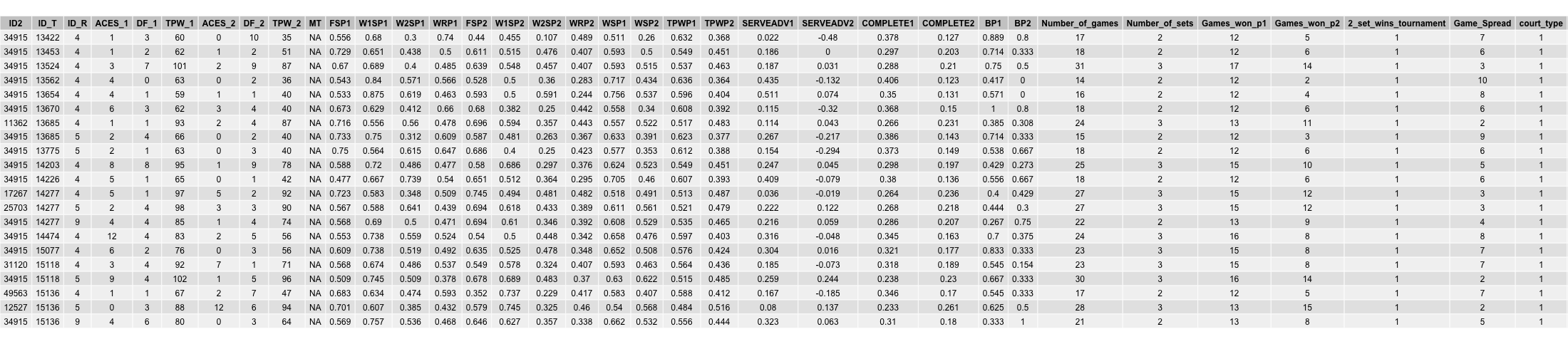


Figure 2 On Court Per Match Statistics of First 50 Matches

 In Figure the number of decimals of decimals is limited to three for readability.

## Feature Construction

After developing metrics representing the qualities of the players in the previous section, the next step is to create the training set for our model. For any given match, we are trying to predict the outcome *pre-game*. Thus we cannot use the metrics for a match as our final features, because we have no access to any of the metrics in our dataset. which will be the case when we will use our model on unseen data. It follows logically that our feature set must be estimated based on the past performance of the players, which is what Knottenbelt did for estimating serve and return winning percentages. If we generalize Knottenbelt’s Common Opponent Model, we can use the subset of past matches of both players where they faced the same opponents and find the average value of our feature for both players. Taking the difference of the same feature between two players, as described in Table 2, will give us the final feature. Knottenbelt and Sipko use the same model, which is shown in Figure 4.  To see the workflow of deriving features, et us look at the match between Pruchya Isarow and Linh Giang Trinh played in Montevideo Challenger 2018[[7]](#footnote-66) as an example. First we extract all the past matches of both of these players. Isarow has 58 past matches and Giang Trinh has 21. shows Giang Trinh’s past matches in our dataset.

 Now we find the players which both players have played against. These players had three common opponents in the past, which is definitely a below average sample size. The number of common opponents is positively correlated with player’s performance estimates and consequently the predictive strength of our model. When we start evaluating our model, the number of past common opponents will be useful metric to evaluate the confidence of our model in the prediction it makes.

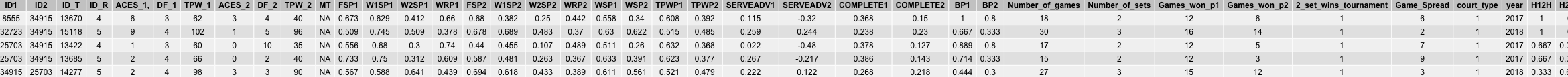
 In order to construct the serve advantage feature for this match (SERVEADV), we average the values from matches both players played against their common opponents and take their difference, as shown in the table below. The final SERVEADV feature for this match will be 0.313.

Table 3 Calculating the SERVEADV feature for Isarow - Trinh

|  |  |  |
| --- | --- | --- |
| Matches | ServeAdv\_Isarow | ServeAdv\_Trinh |
| **Match 1** | 0.387 | -0.32 |
| **Match 2** | 0.316 | 0.244 |
| **Match 3** | -0.094 | -0.48 |
| **Match 4** | NA | -0.217 |
| **Match 5** | NA | 0.222 |
| **Average Serve Advantage** | 0.203 | -0.11 |

## Feature Weigghti

# Tables, Graphics, References, and Labels

## Tables

In addition to the tables that can be automatically generated from a data frame in **R** that you saw in [R Markdown Basics] using the kable function, you can also create tables using *pandoc*. (More information is available at <http://pandoc.org/README.html#tables>.) This might be useful if you don’t have values specifically stored in **R**, but you’d like to display them in table form. Below is an example. Pay careful attention to the alignment in the table and hyphens to create the rows and columns.

Table 1 Correlation of Inheritance Factors for Parents and Child

|  |  |  |
| --- | --- | --- |
| Factors | Correlation between Parents & Child | Inherited |
| Education | -0.49 | Yes |
| Socio-Economic Status | 0.28 | Slight |
| Income | 0.08 | No |
| Family Size | 0.18 | Slight |
| Occupational Prestige | 0.21 | Slight |

We can also create a link to the table by doing the following: Table 1. If you go back to [Loading and exploring data](#loading-and-exploring-data) and look at the kable table, we can create a reference to this max delays table too: Table ??. The addition of the (\#tab:inher) option to the end of the table caption allows us to then make a reference to Table \@ref(tab:label). Note that this reference could appear anywhere throughout the document after the table has appeared.

## Figures

If your thesis has a lot of figures, *R Markdown* might behave better for you than that other word processor. One perk is that it will automatically number the figures accordingly in each chapter. You’ll also be able to create a label for each figure, add a caption, and then reference the figure in a way similar to what we saw with tables earlier. If you label your figures, you can move the figures around and *R Markdown* will automatically adjust the numbering for you. No need for you to remember! So that you don’t have to get too far into LaTeX to do this, a couple **R** functions have been created for you to assist. You’ll see their use below.

In the **R** chunk below, we will load in a picture stored as duke.png in our main directory. We then give it the caption of “Duke logo”, the label of “dukelogo”, and specify that this is a figure. Make note of the different **R** chunk options that are given in the R Markdown file (not shown in the knitted document).

include\_graphics(path = "figure/duke.png")



Figure 7 Duke logo

Here is a reference to the Duke logo: Figure 7. Note the use of the fig: code here. By naming the **R** chunk that contains the figure, we can then reference that figure later as done in the first sentence here. We can also specify the caption for the figure via the R chunk option fig.cap.

Here is a reference to this image: Figure ??.

A table linking these carrier codes to airline names is available at <https://github.com/ismayc/pnwflights14/blob/master/data/airlines.csv>.

Next, we will explore the use of the out.extra chunk option, which can be used to shrink or expand an image loaded from a file by specifying "scale= ". Here we use the mathematical graph stored in the “subdivision.pdf” file.

![Figure 8 Subdiv. graph](data:application/pdf;base64,)

Figure 8 Subdiv. graph

Here is a reference to this image: Figure 8. Note that echo=FALSE is specified so that the **R** code is hidden in the document.

**More Figure Stuff**

Lastly, we will explore how to rotate and enlarge figures using the out.extra chunk option. (Currently this only works in the PDF version of the book.)

![Figure 8 Subdiv. graph](data:application/pdf;base64,)

Figure 9 A Larger Figure, Flipped Upside Down

As another example, here is a reference: Figure 9.

## Footnotes and Endnotes

You might want to footnote something.[[8]](#footnote-79) The footnote will be in a smaller font and placed appropriately. Endnotes work in much the same way.

## Bibliographies

Of course you will need to cite things, and you will probably accumulate an armful of sources. There are a variety of tools available for creating a bibliography database (stored with the .bib extension). In addition to BibTeX suggested below, you may want to consider using the free and easy-to-use tool called Zotero. The Duke librarians have created Zotero documentation at <https://library.duke.edu/research/zotero>. In addition, a tutorial is available from Middlebury College at <http://sites.middlebury.edu/zoteromiddlebury/>.

*R Markdown* uses *pandoc* (<http://pandoc.org/>) to build its bibliographies. One nice caveat of this is that you won’t have to do a second compile to load in references as standard LaTeX requires. To cite references in your thesis (after creating your bibliography database), place the reference name inside square brackets and precede it by the “at” symbol. For example, here’s a reference to a book about worrying: (**???**). This Molina1994 entry appears in a file called thesis.bib in the bib folder. This bibliography database file was created by a program called BibTeX. You can call this file something else if you like (look at the YAML header in the main .Rmd file) and, by default, is to placed in the bib folder.

For more information about BibTeX and bibliographies, see the following documentation from Reed College at (<http://web.reed.edu/cis/help/latex/index.html>)[[9]](#footnote-85). There are three pages on this topic: *bibtex* (which talks about using BibTeX, at <http://web.reed.edu/cis/help/latex/bibtex.html>), *bibtexstyles* (about how to find and use the bibliography style that best suits your needs, at <http://web.reed.edu/cis/help/latex/bibtexstyles.html>) and *bibman* (which covers how to make and maintain a bibliography by hand, without BibTeX, at <http://web.reed.edu/cis/help/latex/bibman.html>). The last page will not be useful unless you have only a few sources.

If you look at the YAML header at the top of the main .Rmd file you can see that we can specify the style of the bibliography by referencing the appropriate csl file. You can download a variety of different style files at <https://www.zotero.org/styles>. Make sure to download the file into the csl folder.

**Tips for Bibliographies**

* Like with thesis formatting, the sooner you start compiling your bibliography for something as large as thesis, the better. Typing in source after source is mind-numbing enough; do you really want to do it for hours on end in late April? Think of it as procrastination.
* The cite key (a citation’s label) needs to be unique from the other entries.
* When you have more than one author or editor, you need to separate each author’s name by the word “and” e.g. Author = {Noble, Sam and Youngberg, Jessica},.
* Bibliographies made using BibTeX (whether manually or using a manager) accept LaTeX markup, so you can italicize and add symbols as necessary.
* To force capitalization in an article title or where all lowercase is generally used, bracket the capital letter in curly braces.

## Anything else?

If you’d like to see examples of other things in this template, please contact Mine Cetinkaya-Rundel (email [mine@stat.duke.edu](mailto:mine@stat.duke.edu)) with your suggestions. We love to see people using *R Markdown* for their theses, and are happy to help.

# Organization

Your paper should be an evolving report on the project in all aspects developed so far, in the form of a draft scientific paper. It must be written in RMarkdown or LaTeX with figures included.

Make sure all figures have font sizes and line widths set so that the final pdf versions are properly legible. Presentation (including correctness of mathematical equations, graphics, tables, citations and bibligraphy, as well as prose) should be pristine.

All details of developments of models, code and examples/analyses must be clearly described – sufficient to that a knowledgeable reader will be able to follow the logic and replicate the analysis.

By the end of the first (Fall) semester, you need to develop a readable interim report. In the second (Spring) semester your task is to evolve this paper into a complete write-up of your work, as if intending to consider submitting to a scientific journal.

However primitive the content may seem to be at the start, start writing.

A fairly standard outline is as follows:

* **Title**
* **Abstract**
* **Chapter 1. Introduction** (setting, problem description, citations, etc.)
* **Chapter 2. Literature review**
* **A Next Chapter:** Some papers have one or two chapters, some papers have several. Keep chapters relatively short: Each section should have one focus. For example,
  + **Chapter 3. New Statistical Models** (theory, ideas)
  + **Chapter 4. Some Computational Issues**
  + **Chapter 5. Simulated Data** (evaluation of models)
  + **Chapter 6. Application** (real motivating problem and data)
  + …
* **Chapter X. Conclusion** (what was done, what was learned, what was good/bad, where research might or could go next)
* **Appendix** (maybe some extra math, details of code)
* **Bibliography** (use bibtex, per the example bib files)

# Conclusion

If we don’t want Conclusion to have a chapter number next to it, we can add the {-} attribute.

**More info**

And here’s some other random info: the first paragraph after a chapter title or section head *shouldn’t be* indented, because indents are to tell the reader that you’re starting a new paragraph. Since that’s obvious after a chapter or section title, proper typesetting doesn’t add an indent there.

# (APPENDIX) Appendix

# The First Appendix

This first appendix includes all of the R chunks of code that were hidden throughout the document (using the include = FALSE chunk tag) to help with readibility and/or setup.

**In the main Rmd file**

# This chunk ensures that the thesisdowndss package is  
# installed and loaded. This thesisdowndss package includes  
# the template files for the thesis.  
if(!require(devtools))  
 install.packages("devtools", repos = "http://cran.rstudio.com")  
if(!require(thesisdowndss))  
 devtools::install\_github("akozlu/thesisdowndss")  
library(thesisdowndss)

**In Chapter 3:**

# This chunk ensures that the thesisdowndss package is  
# installed and loaded. This thesisdowndss package includes  
# the template files for the thesis and also two functions  
# used for labeling and referencing  
if(!require(devtools))  
 install.packages("devtools", repos = "http://cran.rstudio.com")  
if(!require(dplyr))  
 install.packages("dplyr", repos = "http://cran.rstudio.com")  
if(!require(ggplot2))  
 install.packages("ggplot2", repos = "http://cran.rstudio.com")  
if(!require(ggplot2))  
 install.packages("bookdown", repos = "http://cran.rstudio.com")  
if(!require(thesisdowndss)){  
 library(devtools)  
 devtools::install\_github("akozlu/thesisdowndss")  
 }

# The Second Appendix, for Fun

# References

Barnett, T., & Clarke, S. R. (2005). Combining player statistics to predict outcomes of tennis matches. *IMA Journal of Management Mathematics*, *16*, 113–120.

Boulier, B. L., & Stekler, H. O. (1999). Are sports seedings good predictors? An evaluation. *International Journal of Forecasting*, *15*, 83–91.

Chen, Y., Tian, Y., & Zhong, Y. (2017). *Real time tennis match prediction using machine learning* (Master’s thesis). University of Stanford.

Clarke, S., & Dyte, D. (2000). Using official ratings to simulate major tennis tournaments. *International Transactions in Operational Research*, *7*(6), 585–594.

Hertzmann, A., & Zorin, D. (2001). Random walks in tennis. *Missiori Journal of Mathematical Sciences*.

James, O. A. (2008). Probability formulas and statistical analysis in tennis. *Journal of Quantitative Analysis in Sports*, *4*(2), 1–23.

Klaassen, F. J., & Magnus, J. R. (2003). Forecasting the winner of a tennis match. *European Journal of Operational Research*, (148), 257–267.

Newton, P. K., & Keller, J. B. (2005). Probability of winning at tennis i. theory and data. *Studies in Applied Mathematics*, (114), 241–269.

Peters, J. (2017). *Predicting the outcomes of professional tennis matches* (Master’s thesis). University of Edinburgh.

Sipko, M. (2015). *Machine learning for the prediction of professional tennis matches* (Master’s thesis). Imperial College London.

Somboonphokkaphan, A., Phimoltares, S., & Lursinsap, C. (2009). Tennis winner prediction based on time-series history with neural modeling. *IMECS*, *1*, 500–509.

Spanias, A. D., & Knottenbelt, B. W. (2013). Tennis player ranking using quantitative models. *Department of Computing, Imperial College London*.

Webb, P. (2011). Why tennis is big business for bookmakers.

William J. Knottenbelt, D. S., & Madurska, A. M. (2012). A common-opponent stochastic model for predicting the outcome of professional tennis matches. *Computers and Mathematics with Applications*, *64*(17), 3820–3827.

1. Source Code can be found at the author’s [Github](https://github.com/akozlu/TennisPrediction/blob/master/DataExtraction.py) [↑](#footnote-ref-30)
2. The following mathematical calculations are taken from Spanias & Knottenbelt (2013). Derivation of the formulas can be found at Barnett & Clarke (2005) [↑](#footnote-ref-36)
3. The C# source code for implementing Sipko’s paper can be found at [Github](https://github.com/okh1/tennis-prediction) [↑](#footnote-ref-38)
4. In practice, hierarchical models of Barnett show that the difference between the serve-winning probabilities of the two players is enough to predict outcome [↑](#footnote-ref-52)
5. Taken from Sipko & Knottenbelt 3.1 [↑](#footnote-ref-55)
6. You can find the script for extracting metrics and information [here.](https://github.com/akozlu/TennisPrediction/blob/master/FeatureExtraction.py) [↑](#footnote-ref-60)
7. <http://www.tennislive.net/atp/match/pruchya-isarow-VS-linh-giang-trinh/montevideo-challenger-2018/> [↑](#footnote-ref-66)
8. footnote text [↑](#footnote-ref-79)
9. (**???**) [↑](#footnote-ref-85)