Thesis no: MECS-2015-02

Result Prediction by Mining Replays in Dota 2

Filip Johansson, Jesper Wikström

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Game and Software Engineering. The thesis is equivalent to 20 weeks of full-time studies.

Contact Information:
Authors:
Filip Johansson
E-mail: fijo08@student.bth.se

Jesper Wikström
E-mail: jewb08@student.bth.se

University advisor:
Dr. Niklas Lavesson
Dept. Computer Science & Engineering

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Context. Real-time games like Dota 2 lack the extensive mathematical modeling of turn-based games that can be used to make objective statements about how to best play them. Understanding a real-time computer game through the same kind of modeling as a turn-based game is practically impossible.
Objectives. In this thesis an attempt was made to create a model using machine learning that can predict the winning team of a Dota 2 game given partial data collected as the game progressed. A couple of different classifiers were tested, out of these Random Forest was chosen to be studied more in depth.
Methods. A method was devised for retrieving Dota 2 replays and parsing them into a format that can be used to train classifier models. An experiment was conducted comparing the accuracy of several machine learning algorithms with the Random Forest algorithm on predicting the outcome of Dota 2 games. A further experiment comparing the average accuracy of 25 Random Forest models using different settings for the number of trees and attributes was conducted.
Results. Random Forest had the highest accuracy of the different algorithms with the best parameter setting having an average of 88.83% accuracy, with a 82.23% accuracy at the five minute point.
Conclusions. Given the results, it was concluded that partial game-state data can be used to accurately predict the results of an ongoing game of Dota 2 in real-time with the application of machine learning techniques.

Keywords: Dota 2, Machine Learning, Random Forest, Result Prediction

Acknowledgements

We would like to thank our advisor Niklas Lavesson and our families.

Contents

1. Introduction

Real-time games like Dota 2 lack the extensive mathematical modeling of turn-based games that can be used to make objective statements about how to best play them. Discussing, designing, and predicting outcomes of real-time games are therefore mostly done on the basis of experience, expertise, and intuition. Understanding a real-time computer game through the same kind of modeling as a turn-based game is practically impossible, because if specified in extensive form the resulting game tree will be too large to traverse. To put the size of such a tree into perspective consider that the extensive form of chess is too large to be useful [1]. The extensive form of a game is a tree where each node is a game-state and each edge an action leading from one state to another [2]. Chess has an average of 30 possible actions every turn and the longest games are a few hundred turns long. For an extensive form of a real-time game every frame tick on the server would be considered a turn. Considering the longest games being several hours long having hundreds of possible actions for each turn the game tree would be orders of magnitude larger than the one for chess.

In this thesis an attempt was made to create a model that can predict the winning team of a Dota 2[1] game given partial data collected as the game progressed. These predictions could be useful for spectators to understand the current situation of the game or for betting sites to adjust odds in real-time. With this knowledge game developers can determine when a specific game-state indicates a certain end result, and detect imbalances to provide a more interesting and competitive game. An example of knowing the result of a game from a game-state is how modern chess artificial intelligence (AI) uses end-game table bases [3], which dictate optimal play from any game-state with six remaining pieces to all possible end-states.

The attempted method for predicting game outcomes will be to use machine learning, for which a large set of data is required. Replays of publicly available games will serve as the data source. This data will be used to train different machine learning algorithms, resulting in models that can predict the outcome given a partial game state.

2. Related Work

Prediction of game results has been conducted earlier in sports like football and baseball [4]. These predictions were conducted based purely on previous game results and circumstances leading up to the match, such as the weather and the number of injuries. Most machine learning research on real time computer games

[1] `http://dota2.com` *Accessed: 6-1-2015*

has been focused on improving AI by modeling player behavior [5]. Recent work on Dota 2 has been done based on the hero lineups picked in a game to either predict a winner [6], or to give advice on further hero selection[1]. Work has also been conducted on predicting the outcome of single battles instead of the entire game [7].

3. Background

In this chapter different underlying techniques used in this thesis and the mechanics of Dota 2 are explained. From Dota 2 data is gathered and used as the base to train Machine Learning models. Statistical hypothesis tests are used to verify the results acquired from the machine learning models.

3.1 Dota 2

Dota 2 is a stand-alone sequel to the Warcraft 3[2] custom map: Defense of the Ancients (DotA), and is developed by Valve[3].

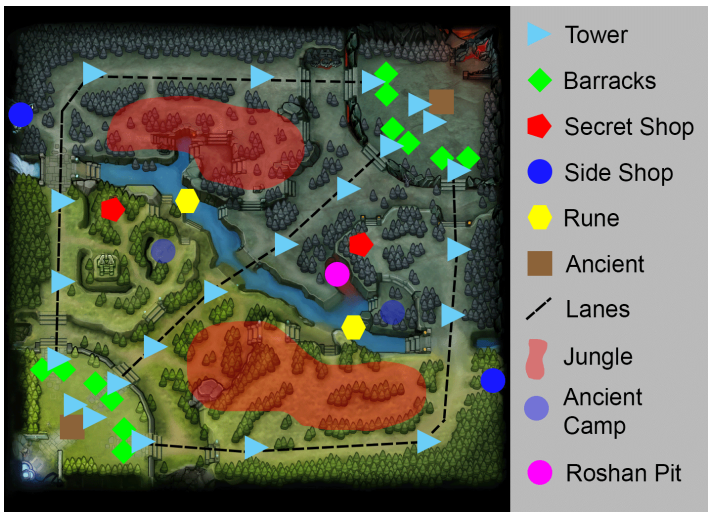

Figure 1: The map of Dota 2, patch 6.74

A game of Dota 2 is played by two opposing teams called Dire and Radiant. It is played on a map divided by a river into two sides (fig. 1), each side belonging to a team. The two teams consist of five players and a number of computer controlled units. Each player controls a unique hero that grows more powerful by purchasing items with gold, and leveling up by gaining experience. If a hero

[1] `http://dota2cp.com` *Accessed: 20-9-2013*

[2] `http://us.blizzard.com/en-us/games/war3/` *Accessed: 6-1-2015*

[3] `http://www.valvesoftware.com` *Accessed: 6-1-2015*

dies, that player will lose an amount of gold and then respawn after a period of time. The amount of gold lost and the time to respawn are both dependent on the hero's current level. Heroes gain experience and gold by killing enemy units. Experience is awarded to all heroes within range whenever an opposing unit is killed. Bounty (gold reward for killing) is only given to the hero that dealt the killing instance of damage (last hit). If a player gets the last hit on their allied units, the bounty will be denied to the opposition and only awarding them a reduced amount of experience.

Figure 2: In-game screenshot from Dota 2 showing the hero Windranger shooting at a creep with a tower shown to the right

Teams have a base of structures situated in opposing corners of the map, each team starts with a number of computer controlled buildings. The victory condition of the game is to destroy the central building in the opponents' base (the Ancient). Three sets of buildings in each base (the Barracks) will spawn small squads of computer controlled units (Lane creeps) every 30 seconds. These units run along fixed paths (Lanes) towards the opposing base, attacking every enemy they encounter. Killing these creeps is the main source of income for both gold and experience, but they are also vital in attacks on enemy structures. Placed along the three lanes each team has three towers with increasing strength closer to the bases. A tower cannot be damaged unless the preceding tower on that lane is destroyed. When a tower is destroyed the opposing team is rewarded with a large bounty. Between the lanes on each team's side there are three (Jungle, Secret shop, Ancient camp) areas whose positions are visualized in fig. 1. The jungle contains groups of neutral units that can be killed for bounty and experience and respawn every minute. The secret shop sells expensive and powerful items. The ancient camp contains a group of neutral units of greater strength with a high

experience and bounty reward upon death. Situated on the Dire side of the river is the Roshan pit, where a very powerful neutral unit called Roshan is located. When Roshan is killed a large bounty is given to the team that got the last hit on him. At the same time an item, the Aegis of the Immortal, is dropped that revives the carrier upon death and is thereby consumed.

Dota 2 has many possible indicators for which team is ahead, either by showing the relative strength of the teams like gold and experience or by showing how close the team is to winning like tower kills. There is however no objective indicator like the score in soccer, since the game ends when either team completes their objective of destroying the Ancient. To find the relationship between the possible indicators and the eventual winner of a Dota 2 game this thesis will employ a systematic approach using machine learning.

3.2 Machine Learning

Machine learning [8] is a field of computer science that governs systems that can learn from data in order to improve performance at a specific task. Within machine learning there is a subsection of problems called supervised learning [8], where example inputs are used to create a model which produces a desired output. The inputs are sets of instances where each instance has a vector of attributes, one of them being the desired output (target attribute). The models are constructed attempting to match the relationship from the attributes to the target attribute found in the data. If the target attribute is a nominal value, it is called the class and the model a classifier. There is a wide variety of machine learning algorithms that due to differences in how they operate construct models that function differently internally, even if they produce similar outputs given the same input. Using several models constructed from the same or different algorithms and then using the mean or mode of those predictions as the final result is a technique to get better predictions called ensemble learning. One such algorithm is Random Forest (RF) [9] that constructs several trees on randomly sampled subsets of instances and attributes.

Validating that the models built are accurately representing the dataset is an important part of machine learning. K-fold cross validation [10] is a model validation technique, which involves splitting the data into k validation datasets. For every subset a model is built on the data not in that subset. The model is then used to predict results from the instances in the validation dataset. The predicted result from the model is compared to the actual result in the data. Different statistical tests can be performed on those results in order to test the models accuracy. Accuracy is the percentage of correct guesses, but it is also important to consider the rates of correct to incorrect guesses for every possible outcome of the model. Evaluating these rates gives a deeper insight of how the model actually performs compared to just looking at the accuracy [11]. The

results will also vary due to random sampling done during the construction of the model, it is therefore important to test the results using statistical tests.

3.3 Statistical Hypothesis Testing

Statistical hypothesis tests are a way of inferring how probable certain hypotheses are from data i.e. how likely is it that a given set of data would arise just from random chance and not because of the proposed relationships in the data. For example, given a set of measurements of two different variables a researcher thinks there's a relationship between the two. To test this hypothesis the researcher calculates the chance that these measurements could occur assuming the null hypothesis is true. The null hypothesis is that there is no relationship between the variables and given the value of one variable the other will be random. The chance of the null hypothesis being true given your data is called the p-value [12]. If the p-value is lower than the significance level, usually 5%, the null hypothesis is rejected as being unlikely to be true which gives further evidence for the hypothetical relationship between the variables.

There are many types of statistical tests designed for different types of hypotheses, analysis of variance (ANOVA) [13] is one such test. ANOVA is performed on a grouping of normally distributed data where the null hypothesis is that the means of all groups are equal. It is used to determine whether there are differences between the groups and if they can be distinguished from each other in the data. For example, given a distribution of cars over their max velocities, a grouping based on chassis color would not create groups with different mean max velocities because there is no relation between the two attributes. This would result in a high p-value in an ANOVA test. Grouping the same distribution based on the cars models would result in differences between the groups and a low p-value from an ANOVA test. ANOVA is designed to work on data that is normally distributed without large differences in the variance of the groups, both of these properties should be tested before using ANOVA to prevent errors in the conclusion. Since ANOVA only tests whether there is any difference between the groups, but not which group or groups are diverging, a common follow up is Tukey's Test [14].

Tukey's Test performs a comparison of means for all possible pairwise combinations of the groups. For each combination it tests if the difference between the two means is less than the standard error. The standard error [15] is the expected difference between the mean of a sampled distribution and the actual mean of the distribution it is sampled from. The Tukey's Test therefore gives the probability of two groups being similar samples of a bigger distribution. To give an example using the same scenario as before with cars, when grouping the cars by model the comparison between two brands of sports cars will give a high p-value in the Tukey's Test since they have similar distributions of max velocities, but

comparing one brand of sports cars with a brand of vans will give a low p-value.

4. Objectives

In this thesis an attempt was made to determine the effectiveness of machine learning techniques with a focus on Random Forest, when applied to classifying the result of Dota 2 games. This was done using partial game-state data that is indicative of success without relying on the specific hero lineups of the teams. To try and prevent the results from becoming irrelevant with future patches, hero lineups were omitted. Patches tend to change hero lineups more than they do item purchases or the importance of objectives on the map. Instead the focus was set on indicators of which team is currently winning for example Hero Kills, Roshan Kills, Items, and more as detailed in Appendix A.

4.1 Research Questions

RQ1 What is the highest average accuracy achievable using different parameters of the Random Forest algorithm?

RQ2 How does the average accuracy of the model vary at different gametimes?

RQ3 What is the relationship between the parameters of the algorithm and the training and validation time?

5. Method

5.1 Data Acquisition

During the initial research no way was found to adequately acquire the quantity of replays needed for the experiments. Dota 2 has an average of 450 000 concurrent players[1] and every match played has a replay temporarily saved on Valve's servers. To get these replays, a way of retrieving a list of recently played games was needed. This was done through Valve's WebAPI[2] described in Appendix C.

In order to download a replay, a so-called replay salt is needed which is not provided through the WebAPI, but can be retrieved through the Dota 2 client. A program based on SteamRE[3] was made in order to retrieve the necessary information. Combining both the WebAPI data and replay salt made the downloads possible.

[1] `http://steamgraph.net/index.php?action=graph&appid=570` *Accessed: 10-4-2013*

[2] `http://dev.dota2.com/showthread.php?t=58317` *Accessed: 6-1-2015*

[3] `https://github.com/SteamRE/SteamKit` *Accessed: 6-1-2015*

Replays of 15146 games were gathered, consisting only of games with the AP (all pick) game mode in the very high skill bracket, played during the period from 2013-02-28 to 2013-04-15 at all times of the day. It was decided to limit the sampling and only include the very high skill bracket to ensure the best quality of play available in public games. It is important to note that no major balance patch[1] was deployed during the replay acquisition period, as such an event might have had an impact on the dataset.

5.2 Data Parsing

Dota 2 game replays are saved in the DEM Format[2], compressed using bzip2 and named after their MatchID, for example "137725678.dem.bz2". Valve does not provide a parser that extracts all the information contained in the replay, which lead the community to produce a number of parsers. Among them is an open source replay parser in Python called Tarrasque[3], that was used to print the network packets stored in the replay file into plain text. This parser was modified to extract the current game-state at each minute. Contributions to Tarrasque were made in order to extract the specific data needed[4].

The output from Tarrasque was converted to the "Raw" format as seen in Appendix A. This was done as parsing replays using Tarrasque was by far the most time consuming part of the entire data processing procedure (there are now newer and faster parsers, one example is Clarity[5], a Java based parser). A simple Python parser was made to convert the raw game-state data into the format, as seen in Team Advantage, Appendix B.

5.3 Preliminary Experiment

The initial hypothesis was that RF would be a good candidate for the dataset because it has shown to be among the best algorithms in other studies [16] [17] [18]. This hypothesis was tested by comparing RF with algorithms from different families. All the classifiers were tested with default settings in R[6] and multiple k-fold cross validation tests were performed with varying seeds. The classifiers were compared based on their accuracy, True Radiant Rate (TRR), and True Dire Rate (TDR). TRR is the rate of correctly classified instances from games where Radiant won and TDR is equivalent for the instances were Dire won. TRR and

[1] `http://dota2.gamepedia.com/Patches` *Accessed: 6-1-2015*

[2] `https://developer.valvesoftware.com/wiki/DEM_Format` *Accessed: 6-1-2015*

[3] `https://github.com/skadistats/Tarrasque` *Accessed: 6-1-2015*

[4] `https://github.com/skadistats/Tarrasque/commits?author=Truth-` *Accessed: 6-1-2015*

[5] `https://github.com/skadistats/clarity` *Accessed: 6-1-2015*

[6] `http://www.r-project.org/` *Accessed: 6-1-2015*

TDR is the sensitivity and specificity [32] of the classification, but are renamed because there is no positive or negative result of a Dota 2 game.

5.3.1 Algorithms

RF operates by constructing multiple decision trees. For each tree a new dataset of equal size to the training set is constructed by random sampling with replacement (bootstrapping [19]) from the original training set. Sampling with replacement means that an instance already added to the subset can be added again, the set will therefore contain some duplicates. During the construction of every tree, each node in the tree only considers a random selection of attributes, choosing the attribute that best splits the set of classes. When making predictions the majority vote from all trees in the model is used. This method of sampling and voting is called bootstrap aggregating or bagging [20].

LibSVM [21] is a library that implements an algorithm from the Support Vector Machine (SVM) [22] family, and was chosen as it had promising results in previous research [23]. SVM works by treating instances of the training set as spatial vectors. It then finds a single or set of hyperplanes that divides up the space between instances of different classes and has the greatest distance to their closest instances. Kernel functions can be used to transform all the instances into a higher dimensional space to find planes between classes, which originally could not be separated in their original space. Classifying new instances is done by putting the instance in that space and assigning it to the same class as the instances on the same side of the hyperplane or hyperplanes. The resulting model is quite complex and hard to decipher due to the nature of the algorithm making SVMs often regarded as black-box solutions.

The Naive Bayes (NB) [24] classifier is often used in problems with text categorization and has proven to be good in earlier empirical studies [25]. Naive Bayes is trained by calculating for each class the mean and variance of the attributes of every instance that belong to that class. An instances predicted class is then the class that has the most probable match given the means and variances calculated in training. The Naive Bayes algorithm thereby creates a simple model that considers all attributes independently when assigning a class, ignoring any co-variance.

The ensemble family encompasses classifiers built on different techniques, bagging and boosting [26]. LogitBoost was chosen has proven to be good in several studies [28], and is a variant of AdaBoost [29] which is a Meta Algorithm that uses multiple simple machine learning algorithms, so called weak learners. LogitBoost [27] is also built around boosting in contrast to RFs bagging. Boosting is a method where all instances of data will have an associated weight. These weights are used by the weak learners when constructing the model (a higher weight is more important to guess correctly). The weights are changed after every iteration, correctly guessed instances gets decreased and incorrectly gets increased.

After a defined amount of iterations the resulting models are combined to make a final complex model.

The last chosen algorithm for this experiment was NNge [30] which comes from the k-Nearest Neighbour (k-NN) algorithm family. NNge is k-NN with generalization and has been used successfully in previous studies [31]. NNge treats instances of the training set as spatial vectors, and generalizes groups of instances with the same class that are close to each other into non-overlapping hyperrectangles. Predicting the class of a new instance is done by finding the class of the nearest instance or hyperrectangle.

5.3.2 Results and Conclusion

During the early stages of evaluation it was discovered that several of the classification algorithms were not suited for the dataset. NNge and LibSVM had training times of over 12 hours on just a subset the dataset and were thus dismissed from future tests. The three remaining classifiers were all tested with two data sources described in Appendix A and Appendix B, with and without correlation-based attribute selection.

Algorithm	Parameters		Accuracy	TRR	TDR
NaiveBayes	None	Team Advg.	85.4%	0.883	0.814
		Raw Data	77.3%	0.963	0.652
	Attri. Selct.	Team Advg.	79.2%	0.837	0.733
		Raw Data	75.8%	0.713	0.950
Random Forest	None	Team Advg.	88.7%	0.886	0.888
		Raw Data	89.4%	0.888	0.903
	Attri. Selct.	Team Advg.	82.4%	0.825	0.823
		Raw Data	78.1%	0.738	0.927
LogitBoost	None	Team Advg.	83.1%	0.838	0.820
		Raw Data	79.2%	0.837	0.733
	Attri. Selct.	Team Advg.	84.1%	0.808	0.915
		Raw Data	78.1%	0.738	0.927

TRR = True Radiant Rate, TDR = True Dire Rate

Table 1: Preliminary Experiment Results

The results in table 1 shows that RF, no selection, with raw data format outperformed the others in average accuracy. Because of this it was decided that future experiments would solely revolve around RF and determining how parameter tuning would change its accuracy.

5.4 Training and Validation Subsets

For each execution of the main experiments, training and validation subsets were constructed by grouping all instances into games by MatchId and randomly sampling 20% of all games into the validation set, with the remaining forming the

training set. This was done in order to ensure that there was no interdependence between instances of the training and validation set. An experiment was performed to make sure that the mean average game-time of the validation sets did not vary too much from the average game-time of the total set of games, as this would be an indication of an error in the sampling method.

The sampling used to create the validation sets was evaluated by comparing the distribution of average game-lengths from generated validation sets to the average game-length of all games. Specifically if the average game-length of all games lay within the confidence interval of the distributions mean average game-length at a confidence level of 0.975.

From 125 generated validation sets the average game-lengths were recorded, these are listed in Appendix E. The standard deviation of those averages and the confidence interval of the mean average game-length for a confidence coefficient of 0.975 is listed in table 2. The average game-length of all games lies within this interval. The specifications of the computer used to run the experiments can be seen in Appendix D.

Label	Value
Standard Deviation	0.125
Confidence Coefficient	0.975
Confidence Interval Min	22.617
Confidence Interval Max	22.649
Average GameLength of all Games	22.632

Table 2: ValidationSet GameTime Data

The distribution of average game-lengths of the validation sets from Appendix E was plotted using a density plot (fig. 3). All the average times fell within twenty seconds of the average game-length of the entire set of games. Because the true mean average game-length of the validation sets lies in the interval given in table 2 with a 97.5% confidence level, there is no indication of an error during the sampling process.

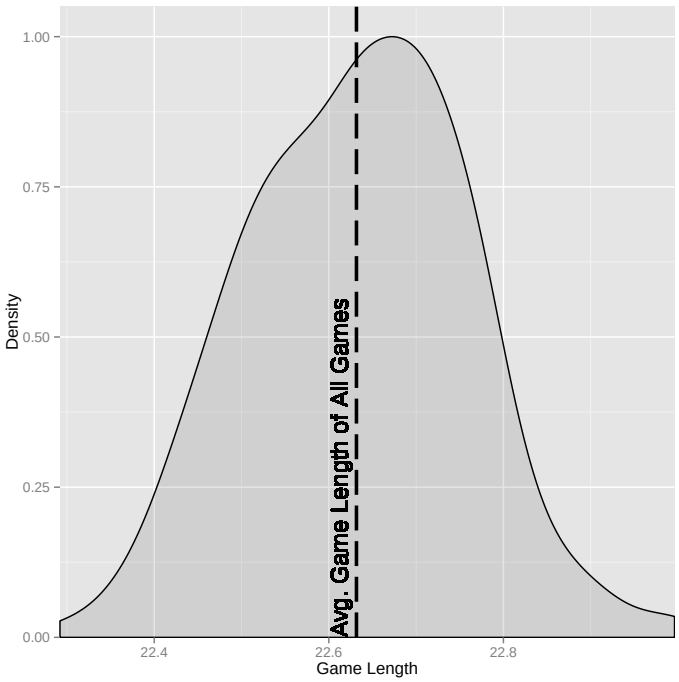

Figure 3: Distribution of Average Game Lengths from Validation Sets

5.5 Experiments

With the focus on RF, Parameter Exploration was performed in order to find the answer to RQ1 and RQ3. The answer to RQ2 was studied in RF Comparisons.

5.5.1 Parameter Exploration

To try and find the optimal parameters for RF, all possible combinations of {1,3,5,7,9} attributes and {10, 60, 110, 160, 210} trees, were trained and validated five times each with new validation and training sets every time. For each parameter combination a measurement of the average accuracy, the time it took to train the model, and the time it took to validate the model was recorded. The parameters were chosen to cover a wide, evenly spaced range.

5.5.2 Random Forest Comparisons

From the results of the Parameter Exploration experiment it was decided to make a more in depth comparison between a parameterized RF with 210 trees 9 attributes, RF with default settings, and a majority class classifier. The default setting for RF in WEKA given dataset the used in this thesis was 100 trees and 8

attributes. The average accuracy of all three models was measured and recorded on a per minute of game-time basis.

5.6 Evaluation

To determine which parameters had the largest impact on accuracy, an ANOVA Test was performed on the resulting distribution of prediction accuracies of all different parameter combinations with the results divided into two groupings by number of trees or number of attributes. Since ANOVA has requirements on the data a Shapiro-Wilk [33] normality test and Levene's Test [34] for homogeneity of variance was performed. Shapiro-Wilk test result was a p-value of 0.9072 and thus the null hypothesis of normality was accepted. Levene's Test resulted in a p-value of 0.9503 and thus the null hypothesis of homogeneity of variance was accepted. Any variance between the means detected would then be further explored with a Tukey's Test to find out which parameters diverged the most from the means of the other groups.

6. Results

The results from the parameter exploration experiment are listed in table 3. The True Radiant Rate (TRR) doesn't improve with different parameters, only True Dire Rate (TDR) performs better. The Validation Time (VT) increases with the number of trees but decreases with the number of attributes. Training Time (TT) scales with both parameters.

Trees	Attributes	Accuracy	TRR	TDR	TT	VT
10	1	86.49%	0.926	0.777	1.96	0.1
10	3	86.91%	0.924	0.790	2.4	0.08
10	5	87.23%	0.923	0.800	2.85	0.08
10	7	87.55%	0.926	0.803	3.39	0.08
10	9	87.50%	0.924	0.804	3.83	0.08
60	1	88.16%	0.925	0.819	11.41	0.36
60	3	88.26%	0.925	0.824	13.78	0.28
60	5	88.44%	0.924	0.828	16.9	0.24
60	7	88.58%	0.923	0.833	20.08	0.22
60	9	88.57%	0.921	0.835	22.98	0.21
110	1	88.41%	0.926	0.825	20.61	0.86
110	3	88.48%	0.925	0.830	25.27	0.6
110	5	88.63%	0.924	0.833	30.92	0.48
110	7	88.61%	0.922	0.836	36.54	0.43
110	9	88.74%	0.922	0.839	42.41	0.4
160	1	88.48%	0.926	0.827	30.47	1.48
160	3	88.58%	0.922	0.833	37.08	1.02
160	5	88.66%	0.922	0.836	44.55	0.82
160	7	88.63%	0.922	0.836	53.45	0.71
160	9	88.78%	0.925	0.836	61.25	0.64
210	1	88.35%	0.924	0.824	38.5	2.07
210	3	88.51%	0.923	0.831	49.06	1.47
210	5	88.66%	0.923	0.835	59.4	1.19
210	7	88.78%	0.922	0.839	70.16	1.02
210	9	88.83%	0.922	0.840	80.03	0.91

TRR = True Radiant Rate, TDR = True Dire Rate,
TT = Training Time, VT = Validation Time

Table 3: Average of Raw Data with Different Parameters, the Training and Validation Time displays average in number of minutes it took to execute

The table 4 contains a summary of the results from the RF comparisons between default parameters, 210 trees 9 attributes, and a majority class classifier. A more detailed summary for each minute of game-time is listed in Appendix F.

Classifier	Instances	Accuracy	TRR	TDR
Majority Class	357984	59.97%	0.600	0.000
Default	355539	87.41%	0.926	0.802
210 Trees - 9 Attributes	359486	88.83%	0.922	0.840

TRR = True Radiant Rate, TDR = True Dire Rate

Table 4: Random Forest Summary

7. Analysis

As seen in fig. 4 the number of trees has the largest positive impact on accuracy. This increase however diminishes with a high number of trees, while an increase in attributes is still contributing to an increase in accuracy. This is corroborated by the ANOVA Test in table 5 and the Tukey's Test in table 6. The p-value is lower for the Trees group than the Attributes group in the ANOVA test but looking at the Tukey's Test 110-210 trees has a higher value than 5-9 attributes, the greatest difference for trees is when increasing from 10 to 60.

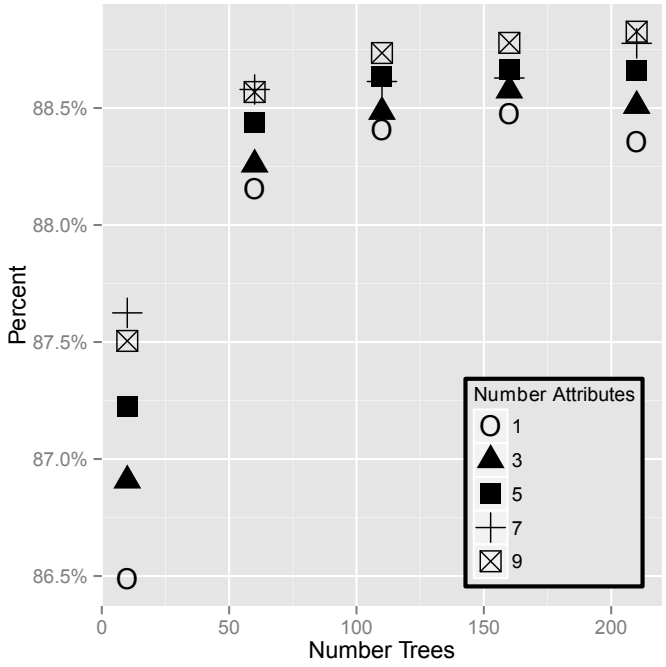

Figure 4: Random Forest Accuracy with different number Trees/Attributes

Variable	n	M	SD	t	p
Trees	4	0.004018	0.0010044	149.35	$<$ 2e-16
Attributes	4	0.000435	0.0001086	16.16	1.59e-10

M = mean, SD = standard deviation

Table 5: ANOVA Test on Parameter Experiment grouped by Trees and Attributes

The time required for building the classifiers scales linearly with either factor and exponentially if you increase both, as can be seen in fig. 5. The validation

	Mean	Bounds		Adjusted		Mean	Bounds		Adjust
Trees	Difference	Lower	Upper	P-Value	Attributes	Difference	Lower	Upper	P-Value
110-10	1.44e-2	1.24e-2	1.65e-2	0.00e+0	3-1	1.69e-3	-3.44e-4	3.72e-3	1.52e-1
160-10	1.49e-2	1.29e-2	1.70e-2	0.00e+0	5-1	3.45e-3	1.42e-3	5.49e-3	6.81e-5
210-10	1.49e-2	1.29e-2	1.70e-2	0.00e+0	7-1	4.54e-3	2.48e-3	6.59e-3	1.00e-7
60-10	1.27e-2	1.06e-2	1.47e-2	0.00e+0	9-1	5.05e-3	3.02e-3	7.08e-3	0.00e+0
160-110	5.03e-4	-1.53e-3	2.54e-3	9.59e-1	5-3	1.76e-3	-2.68e-4	3.80e-3	1.21e-1
210-110	5.03e-4	-1.53e-3	2.54e-3	9.59e-1	7-3	2.85e-3	7.96e-4	4.90e-3	1.82e-3
60-110	-1.74e-3	-3.78e-3	2.88e-4	1.29e-1	9-3	3.36e-3	1.33e-3	5.39e-3	1.13e-4
210-160	-7.45e-7	-2.03e-3	2.03e-3	1.00e+0	7-5	1.08e-3	-9.69e-4	3.14e-3	5.88e-1
60-160	-2.25e-3	-4.28e-3	-2.15e-4	2.24e-2	9-5	1.60e-3	-4.36e-4	3.63e-3	1.96e-1
60-210	-2.25e-3	-4.28e-3	-2.14e-4	2.24e-2	9-7	5.12e-4	-1.54e-3	2.57e-3	9.58e-1

Table 6: Tukey's Test on Parameter Experiment grouped by Trees and Attributes

time was omitted because it made up on average 2% of the total execution time, so the figure more clearly visualizes how the greater contributor to execution time was scaling.

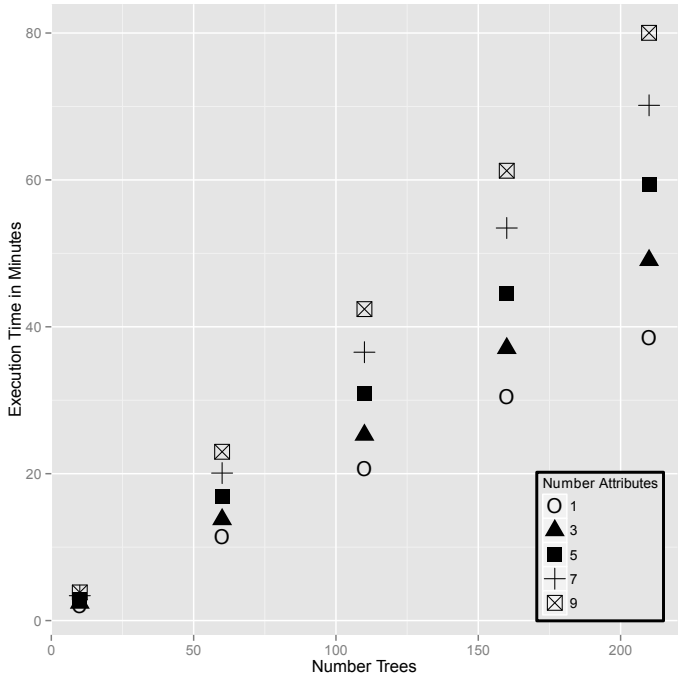

Figure 5: Graph of Execution Time in Minutes required for Building the Random Forest Classifier with different parameters

Figure 6 plots the percentage of games that have not finished yet, which shows how the classifiers become unreliable at higher game-times due to the lack of instances. A high accuracy at early game-times is important to cover most instances. The total area under each curve of correctly classified instances can be

deceiving because the number of instances at every minute drops quickly after 20. By plotting the product of accuracy and "unfinished games", the area under that plot becomes equal to the number of correctly classified instances (as shown in fig. 7), which better shows how many instances are correctly classified per minute.

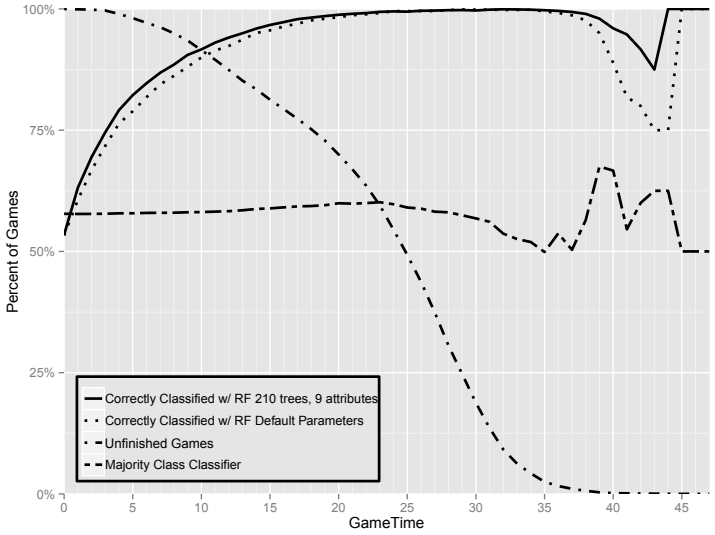

Figure 6: Comparison of Random Forest with two different Parameters, Majority Class and a line representing the number of unfinished games

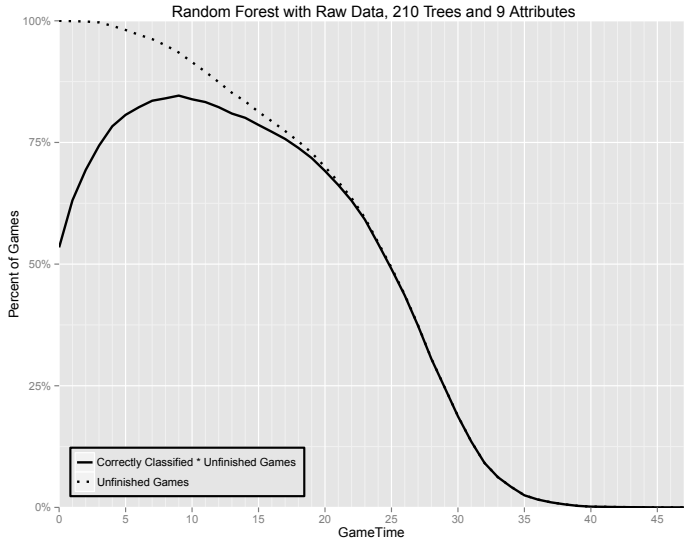

Figure 7: Random Forest accuracy scaled with the amount of unfinished yet

As seen in table 3 the TRR doesn't vary much between different parameters in the RF classifiers and most increase in accuracy is due to the TDR. In fig. 8

this gap between the TDRs of the parameterized and default version of RF can be clearly seen, and it also seems to suggest that the unreliability at high game-times is also due to TDR. There is not much that differentiates Dire from Radiant, so this result is unexpected and no explanation for it was found.

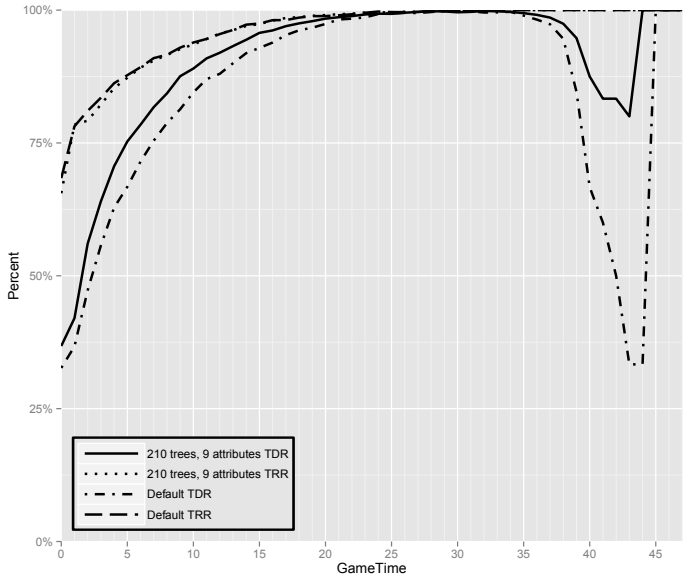

Figure 8: True Radiant Rate and True Dire Rate graphed from Random Forest with default and 210 trees with 9 attributes

8. Conclusions

In this thesis many differently parameterized versions of the RF classification algorithm were used to predict Dota 2 results using partial gamestate data without hero lineups. The results achieved were better than expected but there is a lot more that could have been done.

As seen in table 3 the best model had an average accuracy of 88.83%, which is higher than expected because it was thought that there would be less of a difference between the teams during the first 5 minutes to be used as a basis for classification. However, as seen in fig. 6, the model is already correctly predicting 82.23% of instances at that point. Due to a shortage of long games in the replay set, it is hard to make any conclusive statements regarding the model's effectiveness on games longer than 35 minutes.

Given the results, it was concluded that partial game-state data can be used to accurately predict the results of an ongoing game of Dota 2 in real-time with the application of machine learning techniques. The execution time scaling when training the model leaves room for a lot of possible improvements on the param-

eters of the model or the format of the data used.

9. Future Work

The results from the parameter exploration experiment were not sufficient to conclude that a maximum of accuracy had been found in the parameter space used. Due to time constraints the parameter space could not be expanded in this thesis. Further testing with higher trees and attributes than 210 and 9 could yield higher accuracy.

A higher accuracy could be achieved by using more data from replays, for example: "Stun Duration on Enemies", "Healing Done/Received", or "Damage Dealt to Heroes" etc. Even though hero lineups were intentionally not used in this thesis it would probably increase accuracy as it has been successfully used to predict results in previous works [6].

A possible improvement on the dataset would be to only use replays from professional Dota 2 games, as the quality of play in those games would be higher than those used in this thesis. The amount of such replays that are available is much lower, and if the sampling is restricted to a single patch, it might be difficult to acquire a large number of them. The replay set also lacked long games and introducing a sampling bias towards longer games could have helped in that aspect. About 1.5% of the downloaded replays were broken for unknown reasons, if this error was dependent on aspects of the game it could have caused a slight bias in the sampled dataset.

An understanding as to why there is such a discrepancy between True Dire Rate and True Radiant Rate in table 3 was not reached in this thesis. There are differences between the teams in Dota 2, but they are not great enough that this result was expected.

Random Forest yielded the highest accuracy of the classifiers tested and became therefore the focus of this thesis. The downside to this was that it is one of the classifiers that are difficult to visualize and interpret. Using a classifier that results in a more easily understood model could prove to be useful for understanding the balance of the game. Another way of improving the ease of interpretation of the model would be to reduce the number of attributes using attribute selection.

References

[1] A. Perea, *Rationality in Extensive Form Games*, 6th ed. Springer, 2001.

[2] R. Aumann and S. Hart, *Handbook of Game Theory with Economic Applications*, 1st ed. Elsevier, 1992, vol. 1.

[3] R. Bellman, "On the application of dynamic programing to the determination of optimal play in chess and checkers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 53, no. 2, p. 244, 1965.

[4] A. Joseph, N. E. Fenton, and M. Neil, "Predicting football results using bayesian nets and other machine learning techniques," *Knowledge-Based Systems*, vol. 19, no. 7, pp. 544–553, 2006.

[5] G. Synnaeve and P. Bessière., "A bayesian model for plan recognition in rts games applied to starcraft," in *Artificial Intelligence and Interactive Digital Entertainment*, 2011.

[6] K. Conley and D. Perry, "How does he saw me? a recommendation engine for picking heroes in dota 2," 2013.

[7] P. Yang, B. Harrison, and D. L. Roberts, "Identifying patterns in combat that are predictive of success in moba games," in *Proceedings of the 9th International Conference on the Foundations of Digital Games*, 2014.

[8] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. MIT press, 2012.

[9] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *International Joint Conference on Artificial Intelligence*, vol. 14, no. 2, 1995, pp. 1137–1145.

[11] D. M. Powers, "Evaluation: from precision, recall and f-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[12] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

[13] R. A. Fisher, "On the "probable error" of a coefficient of correlation deduced from a small sample," *Metron*, vol. 1, pp. 3–32, 1921.

[14] D. C. Montgomery, *Design and analysis of experiments*, 8th ed. John Wiley & Sons, 2012.

[15] B. S. Everitt, *The Cambridge dictionary of statistics*. Cambridge University Press, 2002.

[16] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, 2007, pp. 60–69.

[17] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.

[18] R. Caruana, N. Karampatziakis, and A. Yessenalina, "An empirical evaluation of supervised learning in high dimensions," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 96–103.

[19] B. Efron, "Bootstrap methods: another look at the jackknife," *The annals of Statistics*, pp. 1–26, 1979.

[20] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[21] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[23] D. Meyer, F. Leisch, and K. Hornik, "The support vector machine under test," *Neurocomputing*, vol. 55, no. 1, pp. 169–186, 2003.

[24] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 1995, pp. 338–345.

[25] I. Rish, "An empirical study of the naive bayes classifier," in *International Joint Conference on Artificial Intelligence workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[26] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[27] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[28] R. A. McDonald, D. J. Hand, and I. A. Eckley, "An empirical comparison of three boosting algorithms on real data sets with artificial class noise," in *Multiple Classifier Systems*, 2003, pp. 35–44.

[29] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Second European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[30] B. Martin, "Instance-based learning: nearest neighbour with generalisation," Master's thesis, University of Waikato, 1995.

[31] B. G. Weber and M. Mateas, "A data mining approach to strategy prediction," in *IEEE Conference on Computational Intelligence and Games*, 2009, pp. 140–147.

[32] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[33] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[34] I. Olkin, *Contributions to probability and statistics: essays in honor of Harold Hotelling*. Stanford University Press, 1960.

A. Raw Data Format

The format of the file is WEKAs ARFF format[1], and the data consists of 177 attributes. Every attribute is team based so data that is player specific is summed for all players on that team. With the exception of MatchID, GameTime, and Winner all attributes are paired, one for each team. For example, player kills are recorded in two variables, Kills_radiant and Kills_dire. The entirety of the format consists of the following attribute sets for each team: kills, deaths, assists, last hits, denies, runes used, net worth, tower kills, barracks destroyed, ancient hp, roshan kills, and 76 attributes of items constructed from recipes. The majority of items that are purchased without being built from recipes are usually acquired only to eventually finish a recipe or sell later, and are therefore not as indicative of item progression and omitted from the format. MatchID is only used when constructing subsets of the instances, and removed before used in training or validation of models.

B. TeamAdvantage Format

TeamAdvatange is a reparsing of the raw data file where every pair of attributes related to the teams gets converted to a single attribute with the possible values of Radiant, Dire, Initial, or Tie. It will either be Radiant or Dire if one of those teams is in the lead, otherwise it will be initial if the attributes are unchanged or tie if both teams are equal.

Example raw data: @attribute Kills_radiant integer, @attribute Kills_dire integer. Gets converted into: @attribute KillsLeader Radiant, Dire, Initial, Tie.

C. Valve API usage

The url used was:
`https://api.steampowered.com/IDOTA2Match_570/GetMatchHistory/V001/?key=<key>&skill=3&min_players=10&game_mode=1`
In this URL the <key> is personal value which is acquired after accepting Valve's TOS[2], "skill=3" is used to filter to very high skill games, "min_players=10" is used to filter out games where the teams are not completely full and "game_mode=1" is used to filter so the games retrieve are of the game mode "All Pick".

[1] `http://weka.wikispaces.com/ARFF` *Accessed: 6-1-2015*
[2] `http://steamcommunity.com/dev/apiterms` *Accessed: 6-1-2015*

D. Computer Specifications used for time measurements

The computer used for the time measurements can be seen below:

CPU	AMD Turion II Dual-Core 64 bit 2 MB, 2.2GHz
Memory	Kingston 16GB 1333MHz DDR3 ECC CL9
Harddrive	250 GB, 7200 RPM, SATA
Operating System	FreeBSD

E. List of Average Game Lengths of Validation Sets

The table presents a list of all average game lengths from 125 generated validation sets in minutes.

22.45	22.74	22.63	22.61	22.59
22.56	22.42	22.59	22.90	22.76
22.47	22.48	22.70	22.77	22.73
22.78	22.49	22.48	22.80	22.73
22.63	22.79	22.62	22.60	22.79
22.55	22.51	22.69	22.76	22.58
22.64	22.55	22.62	22.65	22.84
22.57	22.51	22.52	22.74	22.76
22.46	22.75	22.65	22.62	22.69
22.63	22.75	22.65	22.69	22.78
22.72	22.52	22.56	22.66	22.71
22.74	22.76	22.67	22.70	22.52
22.69	22.65	22.55	22.72	22.66
22.70	22.84	22.58	22.61	22.68
22.36	22.68	22.47	22.71	22.51
22.68	22.43	22.55	22.69	22.83
22.54	22.63	22.55	22.82	22.53
22.65	22.57	22.67	22.34	22.70
22.55	23.00	22.77	22.72	22.48
22.56	22.65	22.54	22.43	22.75
22.41	22.68	22.56	22.72	22.75
22.46	22.63	22.60	22.66	22.62
22.59	22.61	22.87	22.50	22.67
22.76	22.45	22.77	22.41	22.55
22.73	22.71	22.62	22.66	22.42

Table 7: Average Game Length of Validation Sets

F. Random Forest Per Minute Summary

Default Parameters

GT	Inst.	Acc.	TRR	TDR
0	15004	53.29%	68.38%	32.66%
1	14988	60.63%	78.14%	36.71%
2	14983	66.78%	81.04%	47.28%
3	14956	71.77%	83.51%	55.72%
4	14852	76.38%	86.24%	62.84%
5	14732	78.90%	87.75%	66.75%
6	14584	81.79%	89.25%	71.50%
7	14457	84.44%	90.97%	75.43%
8	14278	86.20%	91.61%	78.75%
9	14051	88.08%	92.94%	81.34%
10	13778	89.98%	93.91%	84.53%
11	13497	91.41%	94.58%	87.01%
12	13206	92.37%	95.49%	88.02%
13	12867	93.67%	96.27%	90.00%
14	12571	95.05%	97.27%	91.90%
15	12239	95.59%	97.47%	92.89%
16	11978	96.39%	98.11%	93.90%
17	11689	96.97%	98.12%	95.29%
18	11336	97.62%	98.59%	96.20%
19	10970	97.99%	98.87%	96.69%
20	10560	98.29%	98.88%	97.40%
21	10126	98.71%	99.03%	98.23%
22	9616	98.89%	99.22%	98.39%
23	8973	99.15%	99.50%	98.63%
24	8184	99.55%	99.73%	99.27%
25	7424	99.54%	99.64%	99.41%
26	6606	99.56%	99.61%	99.49%
27	5610	99.57%	99.66%	99.45%
28	4669	99.76%	99.85%	99.64%
29	3753	99.89%	99.95%	99.81%
30	2864	99.86%	99.88%	99.84%
31	2089	99.86%	99.91%	99.78%
32	1435	99.79%	100.00%	99.55%
33	978	99.80%	100.00%	99.57%
34	670	99.85%	100.00%	99.69%
35	387	99.48%	100.00%	98.97%
36	242	99.17%	100.00%	98.21%
37	151	98.68%	100.00%	97.33%
38	85	97.65%	100.00%	94.59%
39	40	95.00%	100.00%	84.62%
40	18	88.89%	100.00%	66.67%
41	11	81.82%	100.00%	60.00%
42	10	80.00%	100.00%	50.00%
43	8	75.00%	100.00%	33.33%
44	8	75.00%	100.00%	33.33%
45	2	100.00%	100.00%	100.00%
46	2	100.00%	100.00%	100.00%
47	2	100.00%	100.00%	100.00%

Majority Class

GT	Inst.	Acc.	TRR	TDR
0	15132	58.91%	58.91%	0.00%
1	15117	58.90%	58.90%	0.00%
2	15100	58.90%	58.90%	0.00%
3	15057	58.97%	58.97%	0.00%
4	14957	58.99%	58.99%	0.00%
5	14836	58.96%	58.96%	0.00%
6	14681	59.16%	59.16%	0.00%
7	14558	59.23%	59.23%	0.00%
8	14356	59.21%	59.21%	0.00%
9	14129	59.32%	59.32%	0.00%
10	13856	59.45%	59.45%	0.00%
11	13599	59.53%	59.53%	0.00%
12	13295	59.78%	59.78%	0.00%
13	12961	60.03%	60.03%	0.00%
14	12663	60.21%	60.21%	0.00%
15	12334	60.40%	60.40%	0.00%
16	12064	60.57%	60.57%	0.00%
17	11748	60.91%	60.91%	0.00%
18	11431	61.20%	61.20%	0.00%
19	11099	61.51%	61.51%	0.00%
20	10663	61.73%	61.73%	0.00%
21	10231	61.86%	61.86%	0.00%
22	9703	62.02%	62.02%	0.00%
23	9074	62.00%	62.00%	0.00%
24	8275	61.89%	61.89%	0.00%
25	7461	61.39%	61.39%	0.00%
26	6630	61.10%	61.10%	0.00%
27	5651	60.31%	60.31%	0.00%
28	4652	60.04%	60.04%	0.00%
29	3735	59.14%	59.14%	0.00%
30	2871	58.76%	58.76%	0.00%
31	2072	59.46%	59.46%	0.00%
32	1421	58.13%	58.13%	0.00%
33	973	56.01%	56.01%	0.00%
34	666	55.86%	55.86%	0.00%
35	374	53.21%	53.21%	0.00%
36	245	57.55%	57.55%	0.00%
37	137	61.31%	61.31%	0.00%
38	86	63.95%	63.95%	0.00%
39	41	78.05%	78.05%	0.00%
40	21	85.71%	85.71%	0.00%
41	14	78.57%	78.57%	0.00%
42	8	62.50%	62.50%	0.00%
43	4	25.00%	25.00%	0.00%
44	3	33.33%	33.33%	0.00%
45	0	NA	NA	0.00%
46	0	NA	NA	0.00%
47	0	NA	NA	0.00%

Random Forest 210 trees, 9 attributes

GT	Inst.	Acc.	TRR	TDR
0	15231	53.44%	65.53%	36.79%
1	15219	63.14%	78.48%	42.03%
2	15209	69.45%	79.19%	56.06%
3	15181	74.58%	82.24%	64.02%
4	15073	79.15%	85.31%	70.62%
5	14943	82.23%	87.26%	75.28%
6	14793	84.68%	89.17%	78.44%
7	14658	86.84%	90.51%	81.74%
8	14464	88.52%	91.56%	84.33%
9	14238	90.53%	92.66%	87.56%
10	13937	91.66%	93.55%	89.00%
11	13636	93.05%	94.56%	90.92%
12	13313	94.10%	95.60%	91.98%
13	12977	94.99%	96.23%	93.23%
14	12709	95.95%	97.01%	94.42%
15	12380	96.70%	97.37%	95.72%
16	12085	97.28%	98.03%	96.18%
17	11781	97.89%	98.52%	96.98%
18	11457	98.21%	98.69%	97.50%
19	11098	98.52%	98.92%	97.92%
20	10659	98.80%	99.07%	98.39%
21	10193	99.00%	99.22%	98.66%
22	9684	99.14%	99.32%	98.87%
23	9064	99.39%	99.53%	99.18%
24	8293	99.49%	99.60%	99.33%
25	7505	99.44%	99.53%	99.30%
26	6659	99.62%	99.70%	99.51%
27	5703	99.67%	99.68%	99.65%
28	4661	99.74%	99.71%	99.79%
29	3763	99.76%	99.77%	99.75%
30	2861	99.69%	99.75%	99.59%
31	2067	99.85%	100.00%	99.67%
32	1392	99.93%	100.00%	99.84%
33	949	99.89%	100.00%	99.77%
34	643	99.84%	100.00%	99.67%
35	380	99.74%	100.00%	99.43%
36	248	99.60%	100.00%	99.11%
37	160	99.38%	100.00%	98.59%
38	98	98.98%	100.00%	97.44%
39	50	98.00%	100.00%	94.74%
40	25	96.00%	100.00%	87.50%
41	19	94.74%	100.00%	83.33%
42	12	91.67%	100.00%	83.33%
43	8	87.50%	100.00%	80.00%
44	5	100.00%	100.00%	100.00%
45	1	100.00%	NA	100.00%
46	1	100.00%	NA	100.00%
47	1	100.00%	NA	100.00%

Table 8: Summary of Random Forest classifier per minute with different parameters