

# Application of machine learning to help AI to play Hearthstone

Evgeny Patekha

Russia

Email: evgeny.patekha@gmail.com

**Abstract** — This paper presents a solution, which was developed as a part of the competition AAIA'17 Data Mining Challenge: Helping AI to Play Hearthstone. The goal of the competition was to predict the probability of AI player win in different intra-game states of Hearthstone game (online computer game with cards). This solution got the third place at the final leaderboard. The paper describes models and local validation approach, which was very useful for models development without overfitting.

## I. INTRODUCTION

**H**EARTHSTONE: Heroes of Warcraft [1] is a free-to-play online video game developed and published by Blizzard Entertainment [2]. It is a turn-based collectible card game between two opponents, who use constructed decks of thirty cards along with a selected hero with a unique power. Players use mana points (money equivalent) to cast spells or summon minions (units for battle) to attack the opponent, with the goal to reduce the opponent's health to zero.

Author did not play this game before competition, but read the game rules and wiki [3] when developed this solution.

AAIA'17 Data Mining Challenge is the fourth data mining competition organized within the framework of International Symposium Advances in Artificial Intelligence and Applications [4].

For the purpose of this challenge, organizers simulated a large number of Hearthstone gameplays. The task for participants of this competition was to construct a prediction model that can learn how to evaluate accurately particular intra-game states. These models would help to improve AI to play the game of Hearthstone: Heroes of Warcraft.

The paper contains:

- the short description of a competition, data provided to the competitors and the evaluation method that was applied to submitted models
- the description of validation scheme which was used for the models development
- the description of models and main features
- the final results of competition

## I. COMPETITION TASK

The ability to assess accurately a winning chance in different game states is substantial for designing efficient and challenging AI players in many games. In this data mining challenge, participants worked to develop a prediction model for a popular game Hearthstone: Heroes of Warcraft - a collectible card video game developed and published by Blizzard Entertainment.

The data for the competition was generated by the simulation of games between weak AI players. Ideas and models from this competition could be used to improve AI play.

The detailed information about the competition can be found in [5].

### A. Data

The data for this competition were provided in two different formats: JSON and tabular. I worked with JSON files as they contained more information than tabular ones. Files with train data contained information about condition of each of the competing heroes, played minion cards, cards in the hand of the first player (it is assumed that the first player always starts the game) and other features.

The “decision” was a target variable to predict with values ‘1’ if the first player won the game and ‘0’ otherwise.

The test data is available in the same format as the training sets, however, there is no information about the “decisions”.

It was allowed to use external knowledge bases about Hearthstone cards.

Initially, the training data contained descriptions of 2,000,000 game states. During the competition test data were replaced by new ones, data from old test became available for training, and full training set appeared to be 3,250,000 records.

New test data contained 750,000 game states.

### B. Evaluation

The participants of the competition were asked to submit likelihoods of winning by the first player.

The submitted solutions were evaluated on-line and the preliminary results were published on the competition leaderboard (public LB). The preliminary score was computed on a subset of the test set, fixed for all participants. It corresponded to approximately 5% of the test data. The final evaluation was done after the completion of the competition having use the remaining part of the test data (final LB). Those results were also published on-line.

The assessment of solutions was done using the Area Under the ROC Curve (AUC) measure.

## II. SOLUTION

The final solution was the mix of the Gradient Boosted Decision Trees and the Neural Net models. All models were developed in R with LightGBM [6] and MXNet [7] libraries. data.table library [8] was used for data processing before training.

### A. Validation

During the competition organizers decided to replace test data, because of the information that different stages of one game could be both in the train and the test sets, and it could lead to inadequate and useless result of the competition.

However, the same problem was actual for local validation while model training. Training records with different stages of the same game led to model overfitting when Gradient Boosting Trees were used for training. To overcome this issue I decided to split data to different folds and tried to put all potential records from one game to one fold.

The only features that could be used for this goal were “hero\_card\_id” of a player and an opponent (each could have 9 different values). Therefore, I split data into 9 folds by the unique combination of “pl.hero\_card\_id” and “op.hero\_card\_id” (total 81 combinations, 9 to each fold). I tried to achieve uniform distribution of data among the folds so that only one kind of “pl.hero\_card\_id” and one kind of “op.hero\_card\_id” were put in each fold.

This solution fixed the problem of overfitting and was very good for local evaluation of created models. The local results of cross-validation (CV) had high correlation with public leaderboard. This kind of CV gave me excellent tool for fine-tuning of my models without overfitting (to choose right features and to find best parameters).

The score of my model dropped less between public and final leaderboards than the scores of other participants in top-10 (Table I). I suppose that was thanks to the good validation scheme.

### B. Features

Based on the initial data many of new features were created and tested by cross-validation. Features were selected to be used in the final models if they improved score with local validation by more than 0.0001, and scores were improved for most of validation folds.

The following features made a major contribution to improving the score (measured by cross-validation):

- Difference between cumulative “attack” (sum of “attack” of a player and his played cards) and cumulative “health” of an opponent (sum of “health” of an opponent and his played cards) and vice versa
- Difference of “health” of a player and an opponent divided by “health” of a player
- Sum of “health” of minions at player’s hand
- Cumulative “attack” gain (compared to base levels) of played cards
- Number of played cards ready to attack

In addition, I used as features a number of cards (separately played and in hand) with specific IDs, specific costs. The most useful of IDs features were features with IDs of spell cards.

The full list of used features listed in the Appendix.

### C. External data

The competition rules allowed to use external data. I used information about cards properties from hearthstonejson.com [9]. For each card, the database contains base information about cost, attack, health plus some additional features like card class, race, faction, collectible and others.

I tried to use additional features in my model. The most of those features did not improve my models. Only feature “number of neutral class cards” (number of cards with neutral class in player’s hand) used in the main final model.

### D. Models

The main model was the Gradient Boosting Decision Trees (GBDT) implemented by the LightGBM library. GBDT is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models of decision trees [10, 11].

The main model had 140 features. The individual score (AUC) of this model was 0.7987 at public LB.

The second model was the LightGBM too. The features were taken from the first model. There was an idea to review the main model features and create another model by adding them one by one from the beginning. As a result, I got another model with only 58 features with score similar to the main model (0.7983 at public LB). The predictions of these models were slightly different. The mix of these models produced a good gain (0.001) on CV and public LB.

The third model was the Neural Network by the MXNet library. I used the same features as in the first model. The model had 2 fully connected hidden layers of 192 units each, ReLu activation with 50% dropout and softmax output [12]. This model had score 0.7980 at public LB and gave moderate improvement when combined with two LightGBM models.

I used grid-search with cross-validation to find the best parameters for each models. The final parameters listed in Table I.

TABLE I  
PARAMETERS OF THE MODELS

Parameters	LGBM 1	LGBM 2	NN
learning_rate	0.03	0.03	0.07
num_leaves	12	12	-
max_depth	4	4	-
feature_fraction	0.8	0.8	-
bagging_fraction	0.75	0.75	-
averaged by CV num iterations	3599	3547	46
batch_size	-	-	32768

### E. Additional model

Based on analysis of the data, I discovered that for the first two turns the outcome of the game is mostly uncertain. A definite outcome is very rare at this stage. I decided to build another model only for first two turns. The idea was to restrict the model by exclusion of usage of some patterns from the next turns. The prediction of this model was scaled and inserted to the prediction of the main model. This approach helped to drop some of the false predictions with moderate improvements of the score.

### F. Other improvements

After replacement of the test data during competition, some features of the new test set became very different from train set. The rules of the game were changed while new test data were collected. The changes affect features “turn”, “op.deck\_count” and “op.hand\_count”.

I tried to reduce those differences and changed train data:

- data with value of the feature “turn” more than 16 were not used in the models because maximum value of the feature “turn” in the new test data was 16;
- all cases where value of the feature “turn” was less than 11 and value of the feature “pl.crystals\_all” was less than value of the feature “turn” were equate to “turn” value;
- for “turn” 1 and 2 values of the feature “op.deck\_count” were increased by 1;
- for “turn” 1 and 2 values of the feature “op.hand\_count” were decreased by 2 and 1 respectively;

These changes led to moderate improvement of the score.

### G. Other steps without success

One of ideas that I tried without success was the idea to swap data between player and his opponent (with missing cards in player’s hand which we did not have for opponent) to get additional data for training. I hope that some different

game states would be good addition to the train data, but this approach did not help to improve the score. I suppose that the reason for this is that we already have enough data for training.

I tried other machine learning technologies, such as xgboost [13], as it was the best of gradient boosting implementation before LightGBM, K-nearest neighbor (KNN) with different number of neighbors and logistic regression, but all of them had worse score and did not improve the score of the main models when I tried ensemble.

I also tried to build second level model to stack different models, as it very popular in many competitions method to improve score, but did not find a way to validate second level model without overfitting. Prediction from stacked model did not improve score at public leaderboard while it was better on my CV.

### H. Training process and final prediction

As we had big training dataset, predictions for the test data were made by each of iterations during 9-folds cross-validation (at the point of the best validation score) and were averaged before submit. This approach had better scores than predictions from single model with full train set with approximation of number of training iteration.

For the final submission all models were trained with 3 different random seeds and the predictions were averaged. This approach is traditional way to increase stability of models.

The final prediction was a weighted mean of models with weights 30% for each of 2 LightGBM models and 40% for the MXNet model. This blend got score 0.8001 at public LB and 0.79895 at final LB (3 place).

## III. FINAL RESULTS

There were submissions from 188 teams from 28 different countries. Top-10 scores are listed in Table II.

I think that the main contribution to my good result was made by a good local validation scheme. My CV allowed me to check many different ideas and to choose the best ones without overfitting the models.

TABLE II  
FINAL RESULTS (TOP-10)

Rank	Participants	Public lb	Final lb	Drop
1	iwannabetheverybest	0,8041	<b>0,80185</b>	-0,0022
2	hieuvq	0,8016	<b>0,79922</b>	-0,0024
<b>3</b>	<b>johnpateha</b>	0,8001	<b>0,79895</b>	<b>-0,0011</b>
4	vz	0,7997	<b>0,79733</b>	-0,0024
5	jj	0,7997	<b>0,79707</b>	-0,0026
6	karek	0,8000	<b>0,79685</b>	-0,0032
7	podludek		<b>0,79657</b>	
8	akumpan	0,7995	<b>0,79654</b>	-0,0030
9	iran-amin		<b>0,79637</b>	
10	basakesin	0,7988	<b>0,79617</b>	-0,0026

## APPENDIX

At the first and the third models were used 140 features(for played IDs used only top important features):

- pl.hero\_card\_id, pl.crystals\_all, pl.crystals\_current,
- pl.hp, pl.armor, pl.attack, pl.special\_skill\_used,
- pl.weapon\_durability,
- pl.deck\_count,
- pl.hand\_count, pl.played\_minions\_count
- op.hero\_card\_id, op.crystals\_all, op.crystals\_current,
- op.hp, op.armor, op.attack, op.special\_skill\_used,
- op.weapon\_durability,
- op.deck\_count,
- op.hand\_count, op.played\_minions\_count
- pl.cum\_attack, pl.cum\_hp\_cur, pl.cum\_attack\_gain,
- pl.cum\_hp\_loss, pl.num\_taunt, pl.num\_can\_attack,
- pl.cum\_crystals\_cost
- op.cum\_attack, op.cum\_hp\_cur,
- op.cum\_attack\_gain,
- op.cum\_hp\_loss,
- op.num\_taunt,
- op.num\_can\_attack, op.cum\_crystals\_cost
- turn, pls.cum\_attac\_hp\_dif, pls.cum\_attac\_hp\_dif1,
- pls.cum\_attac\_hp\_dif2,
- pls.hp\_dif,
- pls.hp\_dif\_to\_hp,
- pl.ids\_count,
- op.ids\_count,
- pl.crystals\_use, op.crystals\_use, pl.cristal\_turn\_dif,
- op.cristal\_turn\_dif
- m\_cum\_attack,
- m\_cum\_hp,
- m\_num\_taunt,
- m\_num\_freezing, m\_cum\_crystals\_cost, s\_num,
- s\_cum\_crystals\_cost,
- w\_attack,
- w\_num,
- w\_cum\_crystals\_cost,
- m\_ids (30), s\_ids (44), pl\_ids (top 3), op\_ids (top 3)

At the second model were used 58 features:

- pl.hero\_card\_id, pl.crystals\_all, pl.hp, pl.armor,
- pl.attack,
- pl.deck\_count,
- pl.hand\_count,
- pl.special\_skill\_used,
- pl.played\_minions\_count,
- pl.max\_cost, pl.avg\_hp\_cur, pl.cum\_attack\_gain,
- pl.cum\_hp\_loss, pl.nocan\_cum\_attack

- op.hero\_card\_id, op.crystals\_all, op.hp, op.armor,
- op.attack,
- op.deck\_count,
- op.hand\_count,
- op.special\_skill\_used, op.played\_minions\_count,
- op.max\_cost, op.avg\_hp\_cur, op.cum\_attack\_gain,
- op.cum\_hp\_loss, "op.nocan\_cum\_attack
- m\_cum\_attack, m\_cum\_hp, m\_num\_neutral\_class,
- s\_cum\_cost
- turn, pls.cum\_attac\_hp\_dif, pls.cum\_attac\_hp\_dif1,
- pls.cum\_attac\_hp\_dif2,
- pls.hp\_dif\_to\_hp,
- pls.cum\_attac\_hp\_dif\_by\_turn,
- pl.crystals\_use,
- op.crystals\_use,
- pl.cristal\_turn\_dif,
- op.cristal\_turn\_dif
- op.cost1, op.cost2, op.cost3, op.cost4, op.cost7,
- m\_cost5, m\_cost6, m\_cost7, s\_cost1, s\_cost2,
- s\_cost3, s\_cost4, s\_cost5, s\_cost6, s\_cost7, pl.cost7

Next prefixes were used in the features names:

- pl. – player played cards
- op. – opponent played cards
- m\_ – minions at player hand
- s\_ – spell at player hand
- w\_ – weapon at player hand

## REFERENCES

- [1] Hearthstone: Heroes of Warcraft. [Online]. Available: <http://eu.battle.net/hearthstone/en/>
- [2] Blizzard Entertainment. [Online]. Available: <http://www.blizzard.com>
- [3] Hearthstone wiki. [Online]. Available: <http://hearthstone.gamepedia.com>
- [4] International Symposium Advances in Artificial Intelligence and Applications. [Online]. Available: <https://fedesis.org/2017/aaia>
- [5] AAI17 Data Mining Challenge. [Online]. Available: <https://knowledgepit.fedesis.org/contest/view.php?id=120>
- [6] Microsoft LightGBM library. [Online]. Available: <https://github.com/Microsoft/LightGBM>
- [7] MXNet library. [Online]. Available: <http://mxnet.io>
- [8] data.table library. [Online]. Available: <http://r-datatable.com>
- [9] hearthstonejson.com. [Online]. Available: <https://api.hearthstonejson.com/v1/18792/enUS/cards.json>
- [10] Jerome H. Friedman, "Greedy function approximation: A gradient boosting machine." Ann. Statist. 29 (2001), no. 5, 1189–1232. doi:10.1214/aos/1013203451 [Online]. Available: <https://statweb.stanford.edu/~jhf/ftp/trebst.pdf>
- [11] Tianqi Chen, Carlos Guestrin, "XGBoost: A Scalable Tree Boosting System" KDD '16, August 13-17, 2016, San Francisco, CA, USA doi:10.1145/2939672.2939785 [Online]. Available: <https://arxiv.org/abs/1603.02754>
- [12] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" Journal of Machine Learning Research 15 (1). 1929-1958 [Online]. Available: <http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf>
- [13] xgboost library. [Online]. Available: <https://github.com/dmlc/xgboost>