Predicting StarCraft II Win Probability

by Jared Lloyd

Executive Summary

StarCraft II (SC2) is a Real-Time Strategy game that has been a popular eSport (competitive on-line game) since 2010. The most current version of the game is named Legacy of the Void, and has been free to play since 2017, leading to a resurgence in popularity. The game has a player base of around 500,000 players worldwide, and professionals have earned as much as 1 million USD in prize money[1].

The goal of the game is to harvest resources, build structures, control an army, and ultimately destroy the opposition players' buildings. A game officially ends when one player has lost all of their buildings, but most games end after a player concedes – because they think (perhaps incorrectly) that there is a low probability of victory. Draws are possible, but these are rare.

Motivation

Since most games end with players conceding, it may be beneficial to have a tool which allows a player to reassess their decision by re-watching the replay and being informed of the statistical probability of their victory at any point.

Players in general invest a lot of effort into understanding how they could improve their gameplay, with some spending as much as $85 an hour for professional coaching[2].

Such a tool could also find applications in broadcasting, where casters of live games could add an overlay showing the probability of victory for each player, adding to the drama and excitement when a "big event" swings the win probabilities. It could also find a use in training AI, as an additional reward metric for AI training[3].

Results

A Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) is trained on game Event Data to have accuracy ranging between 50 and 90% accuracy in predicting game outcome, depending on the fraction of the game it sees. This compares favourably to a secondary model based only on Metadata which has a prediction accuracy of 58%. The **baseline RNN with a window size of 60 (300 seconds) achieved an average accuracy of 76%.**

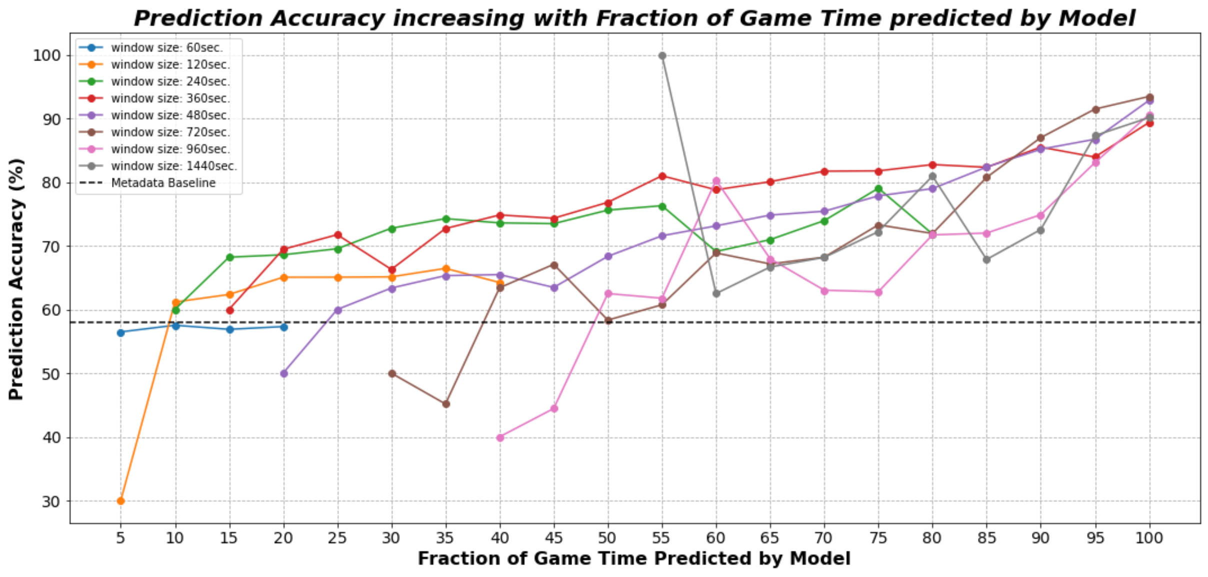

FIGURE 1

[1] https://liquipedia.net/starcraft2/Winnings

[2] https://www.pigstarcraft.com/coaching/

[3] StarCraft II is already a field of interest for AI research, for example Google Deepmind's AlphaStar

1 Methodology

Data are collected from two primary sources. To download the cleaned data used for the modeling process, use this link:

1. A dataset supplied by Blizzard and available for download from their GitHub page – This dataset was constructed with the aim of training AIs to play SC2.
2. User uploaded replays downloaded from SpawningTool – an online repository used for sharing and analyzing replays.

The Blizzard dataset of 1.2 million replays is used to extract Metadata (defined below) to model the outcome of a game at its outset. The SpawningTool dataset of 48,479 replays is used as the training, validation, and test data for the RNN model, which predicts game outcome at various stages of the game, based upon events within the game.

1.1 Definitions

At this point it is beneficial to define terms that are used in this report:

- Metadata – This is information about the game setup, such as the rankings of players, the game length, map name, and races of the players.
- Race – There are 3 races in the game which a player can choose to play: Protoss, Terran, and Zerg.
- Replay file – A replay file is generated every time a game is played. SC2 is a fully deterministic game, meaning that the replay contains only a series of events. Events are recorded every 16th of a second.
- Window size – The number of rows of data starting at time zero. Each row contains an aggregation of 112 frames of events, equivalent to 5 seconds.

1.2 Process

Replays are processed using the sc2reader package (available from Python pip): once to extract Metadata (for the Blizzard and SpawningTool datasets) and then again to extract event data (for SpawningTool only).

Cleaning involves selection of valuable features, discarding or filling of missing values, and transformation of categorical variables into numerical form. The features were assessed by first extracting all attributes from the replay object created by sc2reader, and then reading the definitions of those attributes in the sc2reader code. Forward filling was used for all missing numerical values – e.g., if a player has 300 resources, then NaN values are reported until that value changes. 0 filling was used for all missing categorical values.

For both models the target selected was named “winner” and was either a 0 or 1. 1 represents a victory for Player 1 and 0 a victory for Player 2.

2 Findings

2.1 Metadata

Metadata was modeled using a GridSearchCV to select between K Nearest Neighbors (KNN), Random Forest (RF) and Logistic Regression (LR) models. The goal of this section was to develop the baseline knowledge of game outcome that can be obtained at the beginning of the game. The victory split was balanced, with approximately 51% of value 0 and 49% of 1, so there was no need to perform any data collection.

Estimator agnostic parameters were:

- Different scaling methods, i.e. MinMaxScaler (scales between 0 and 1) and StandardScaler (scales the mean to 0 and the standard deviation to 1).
- Principal Component Analysis (PCA) with a range of components, and with no PCA at all.

There were 3 modeling runs:

1. Grid search with KNN, RF and LR – Using a dataset that contained information on the maps the games were played on.
2. Grid search with KNN, RF and LR – After removing the map data. Removing the map data tended to improve the model accuracy.
3. Grid search with KNN only – After ascertaining that KNN performed best, a more detailed grid search was run to derive the best possible performance.

The best performing model parameters were: 1000 neighbors, with 6 PCA components, and a StandarScaler.

Below is a chart of the performance of all the different models that we tested, showing that KNN reached higher performance than the other estimators, towards the end of modeling. **The best prediction accuracy achieved on the test sets was 58%**.

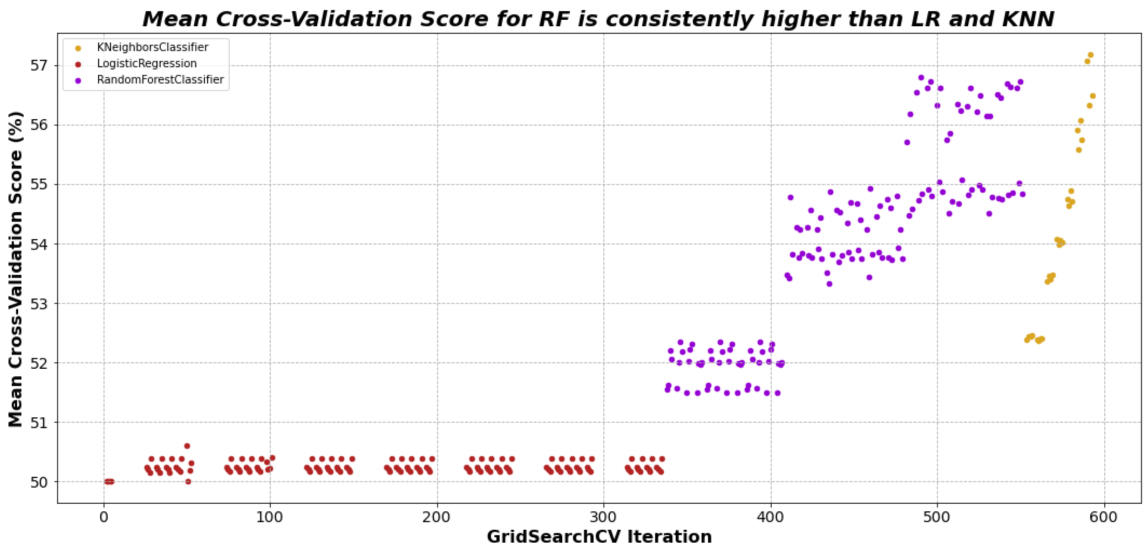

FIGURE 2

2.2 Event Data

An LSTM RNN was created with 1 hidden layer and 128 cells, followed by a Dropout of 0.2 (to prevent over-fitting by randomly dropping cells) and a BatchNormalization layer to avoid the vanishing gradient problem. GRU nodes were tested but found to perform worse in all cases for this dataset.

Memory limitation issues were addressed by using a DataGenerator function to load data from npy files on the hard drive. This allowed modeling on the entire dataset, and so no filtering of Event Data was implemented.

Window sizes of varying length, always starting at the beginning of the game, were used to train the model. The results of this modeling are displayed in **Figure 1** above.

3 Future Work

The model is proof-of-concept that RNN is applicable to predicting the probability of victory in a complex strategy game like SC2. In future work focus should be given to:

- Optimizing model hyperparameters.
- Creating an aggregated model, which contains many models trained on different lengths and periods, dynamically using the most appropriate to make live predictions.
- Investigating the effect of new features that could be created, such as a calculated worker difference between the two players.
- Identifying features that can be removed without impacting model accuracy, to improve model performance and ascertain which features are most predictive of game outcome.