Starcraft II Final Report

Jake Getz

April 2022

By all accounts, achieving a high league in Starcraft 2 online play is hard. In fact players spend months, if not years, wobbling up and down within their league, struggling to break into the next tier. This process is hard enough that it has spawned and supports an entire online infrastructure around strategy guides, coaching, and tips & tricks all targeted at helping players “rank up”. Our goals include finding specific, non-strategic, changes players can make to move up to the next league of play as well as using machine learning to detect potential Smurfing, which is the practice of a high ranking player making a new account and playing in leagues well below their skill level. We are going to examine each league transition, as well as ranking up as a whole, to see which metrics observed in a single game are the best predictors of league, and therefore what someone looking to rank up can focus on beyond strategic decision making to move up in league.

Data

We took our data from the UC Irvine machine learning repository located at: https://archive.ics.uci.edu/ml/datasets/skillcraft1+master+table+dataset . Our data has 3395 entries, each specific to a snapshot of a single player and that single player’s actions in a single game of starcraft II. We have the snapshot statistics, including their League Index as an integer ranging from 1 to 7, the age of the player, the weekly hours that player plays, and the total hours that player has played, each as integers. Our game specific columns cover the actions taken by a player, independent of when or why those actions were taken. They include:

- APM, which stands for actions per minute

- SelectByHotkeys, the number of unit or building selections made by hotkeys per time stamp
- AssignToHotkeys, the number of units or buildings assigned to hotkeys per time stamp
- UniqueHotkeys, the number of unique hotkeys used per time stamp
- MinimapAttacks, the number of attack action made on the minimap per time stamp
- MinimapRightClicks, the number of right clicks made on the minimap per time stamp
- NumberOfPACs, the number of perception action cycles per time stamp (we will define perception action cycles in the following paragraph)
- GapBetweenPACs, the mean duration in milliseconds between PACs
- ActionLatency, the mean latency from onset of a PAC to their first action in milliseconds
- ActionsInPAC, the mean number of actions within each PAC
- TotalMapExplored, the number of 24x24 game coordinate grids viewed by the player per time stamp
- WorkersMade, the number of SCVs, drones, and probes trained per time stamp
- UniqueUnitsMade, the number of unique units made per time stamp
- ComplexUnitsMade, the number of ghosts, infestors, and high templars trained per time stamp
- ComplexAbilitiesUsed, the number of abilities requiring specific targeting instructions used per time stamp.

In the list of attributes in our data we discuss something called a perception action cycle. A perception action cycle, or PAC, consists of a shift of the screen to a new location for some time, followed by at least one action, though often closer to four to six, and then a shift to some other location. Like APM, it can be used to measure how rapidly a player is making choices, but also how easily a player can shift focus from one area of play to another.

Exploratory Data Analysis

When we began exploratory data analysis, we started by charting the league index vs the count of each league index to see where players seemed to group, shown below:

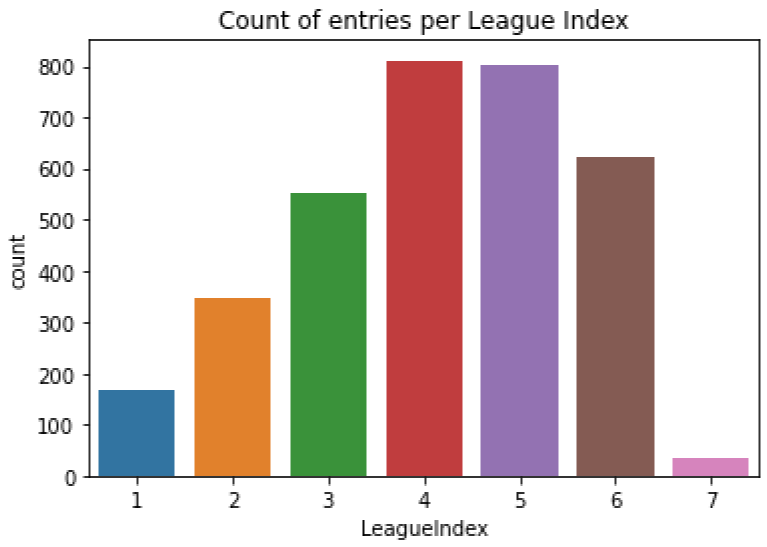

Figure 1. The counts of our data for each league

Figure 1 suggests that most players in our data set trend towards leagues 4 and 5, with not many players being stuck in 1 or 2 and extremely few players achieving the highest league of 7. We then moved on to plotting the league index against other features to see what of note we could find. One of the most surprising plots was Figure 2, showing Total hours against League Index:

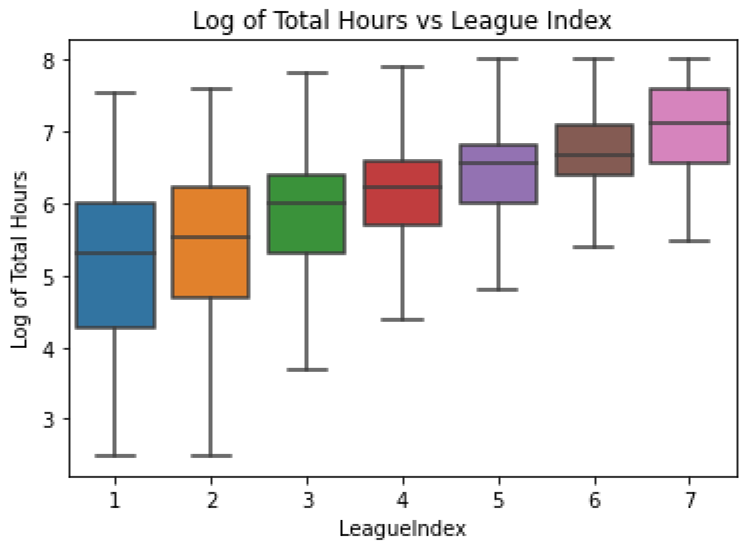

Figure 2. A boxplot of League Index vs. the log of Total Hours

One would expect that total hours would continuously increase across the later leagues, and indeed *Figure 2* shows an exponential increase in total hours, as our figure is a plot of the log of total hours. However *Figure 2* also suggests that increasing total hours will help you increase leagues until you reach the 5th league, but after that we start to see slight diminishing returns as the mean of the log of total hours between leagues 5 and 6 are very similar and the whiskers have nearly identical top whiskers. It is worth noting that while league 7 seems to have a considerably higher mean log of total hours, as *Figure 1* showed, we don't have a huge amount of data about players in the highest league, so the trustworthiness of this data is questionable. On the flipside, APM and League Index were related almost exactly as expected:

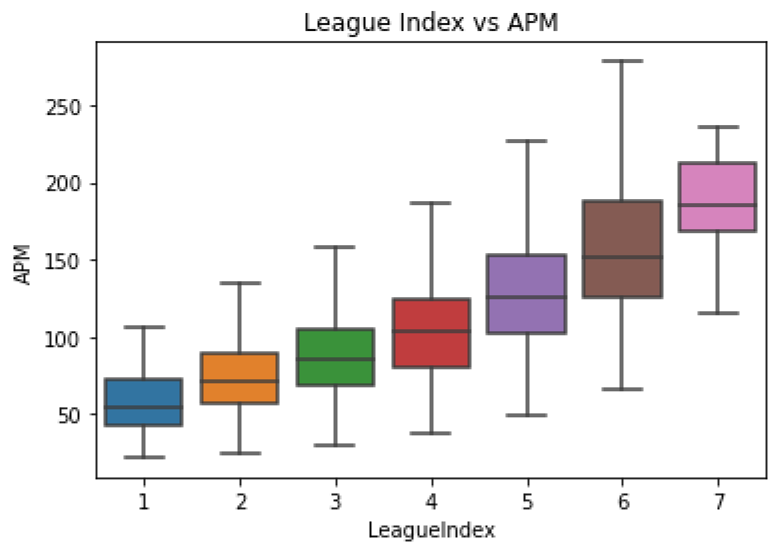

Figure 3. A boxplot of League Index vs APM

Players in higher leagues perform more actions per minute. An unsurprising phenomenon as real time strategy games rely on not only a solid strategic plan but also quick reaction to the things your opponents do. One of the more interesting things we discovered is how higher league players tend to act quickly after a change in focus.

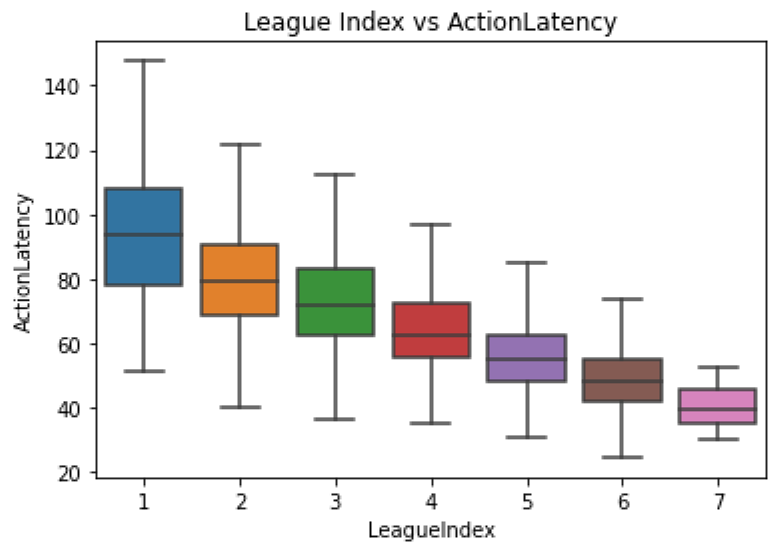

Figure 4. A boxplot of League Index vs Action Latency

We can see that in *Figure 4* as League index increases, action latency decreases rapidly, suggesting that the higher the league the less time it takes a player to react to the new information brought on by a shift in map focus. It is worth noting that the whisker lengths also reduce consistently, reinforcing that these means aren't being artificially lowered by a handful of extremely low values but are in fact generally trending downwards.

Feature Importances

Figure 5 below shows a heatmap of correlation amongst the features. This might help us locate instances of multicollinearity which could potentially impact our measures of feature importance.

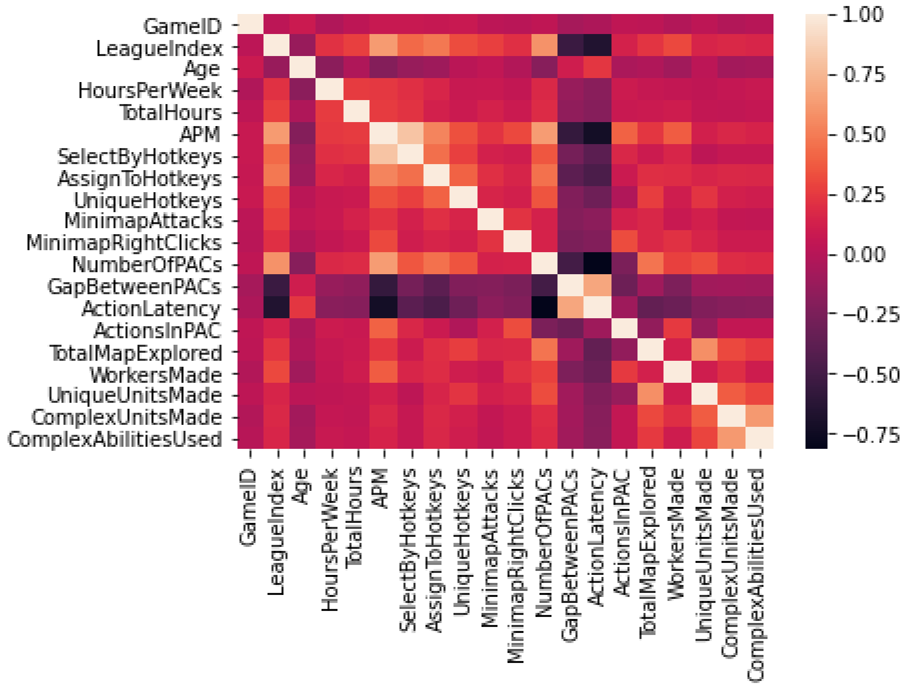

Figure 5. A heatmap of feature correlations

We have a handful of light colored spots in *Figure 5*. Places where our correlation is above .8 we drop one of the two features to lessen the spreading effect of having high correlation during feature importance. In our case that is only the correlation between APM and SelectByHotkeys. We drop SelectByHotkeys for our feature importance analysis.

We initially check feature importance for the entire data set, independent of league transitions as shown in *Figure 6*. These were generated from a Random Forest model after we fit it on the training data.

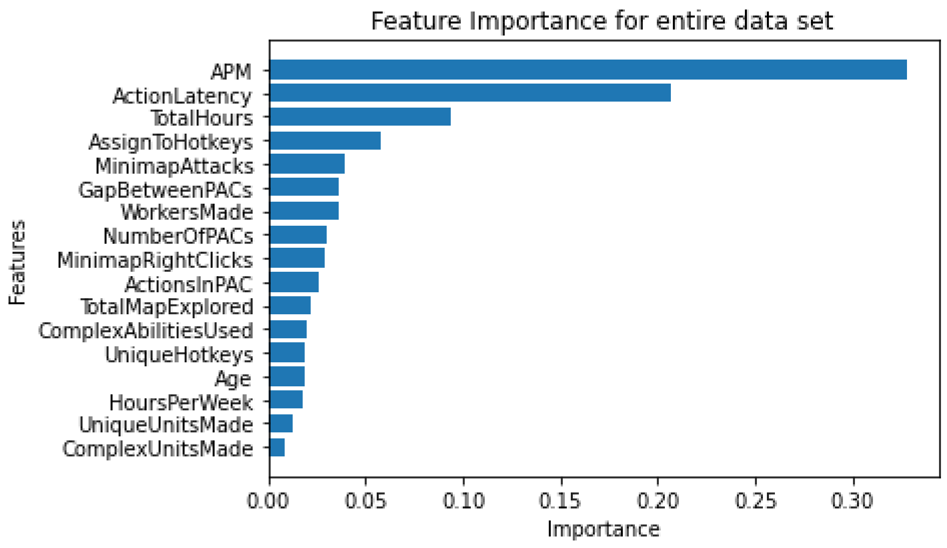

Figure 6. Feature importance for the entire data set found by Random Forest.

Figure 6 suggests that for the entire range of leagues, APM is a powerful predictor of league index, followed by Action Latency after a significant drop, and then Total Hours after another significant drop. We are interested in whether or not this order will stay close to the same for individual league transitions, which we examine below in *Figure 7.*

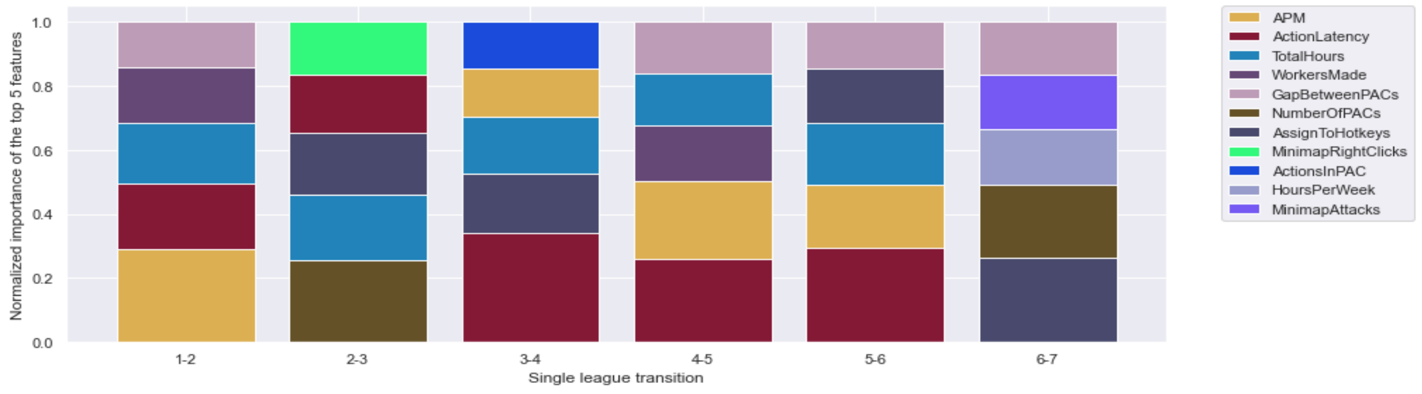

Figure 7. The top 5 feature importances for each league transition by Random Forest normalized to sum to 1

We can see that across most transitions our top 3 importances from *Figure 6* are present in *Figure 7.* Interestingly, while APM was far above Action latency in *Figure 6* we can see that in *Figure 7* after the transition from 1-2 Action Latency is consistently greater than APM, including being present in 2-3 while APM is absent. We can see that none of our expected predictors are present in the 6-7 transition but, as shown in *Figure 1* we don't have a large amount of data from players in league 7 so the accuracy of the transitional data is questionable as players in league 6 will overwhelm data from players in league 7. We repeated feature importance using logistic regression for each transition, and found overall feature importances through shap feature importances in order to confirm our findings from the random forest method.

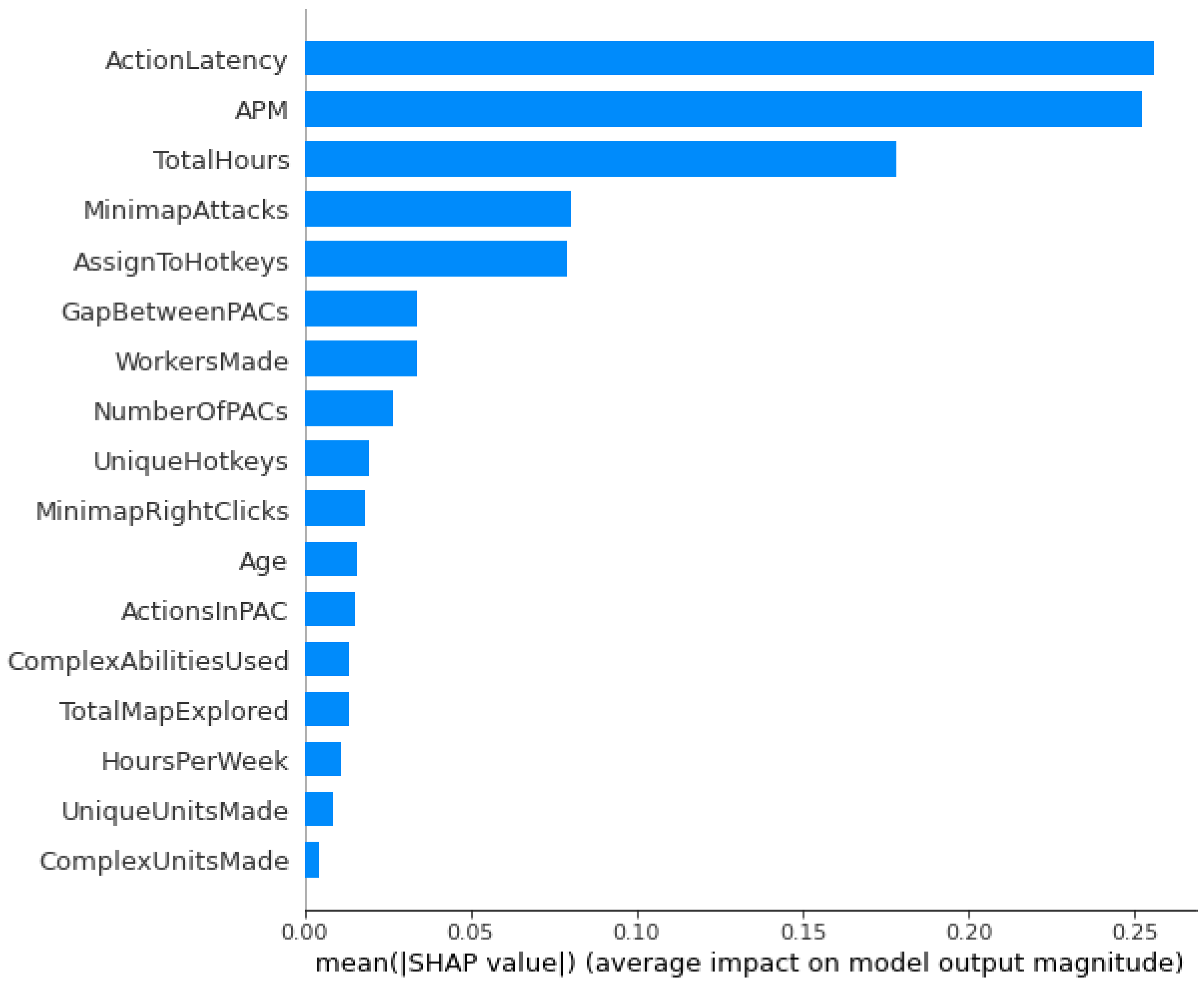

Figure 8. Shap feature importance for the entire data set.

Interestingly, we find that while the top 3 features remain the same in *Figure 8* as they do in *Figure 6* we see Action Latency having slightly more importance than APM with total hours continuing to be a solid third place. When we refer back to *Figure 7* we see that Action Latency is the most important feature for league indices from 3 to 6 which also happens to be where most of our data lies. We want to confirm the feature importances per league transition before we use it definitively, so we check in with another machine learning method.

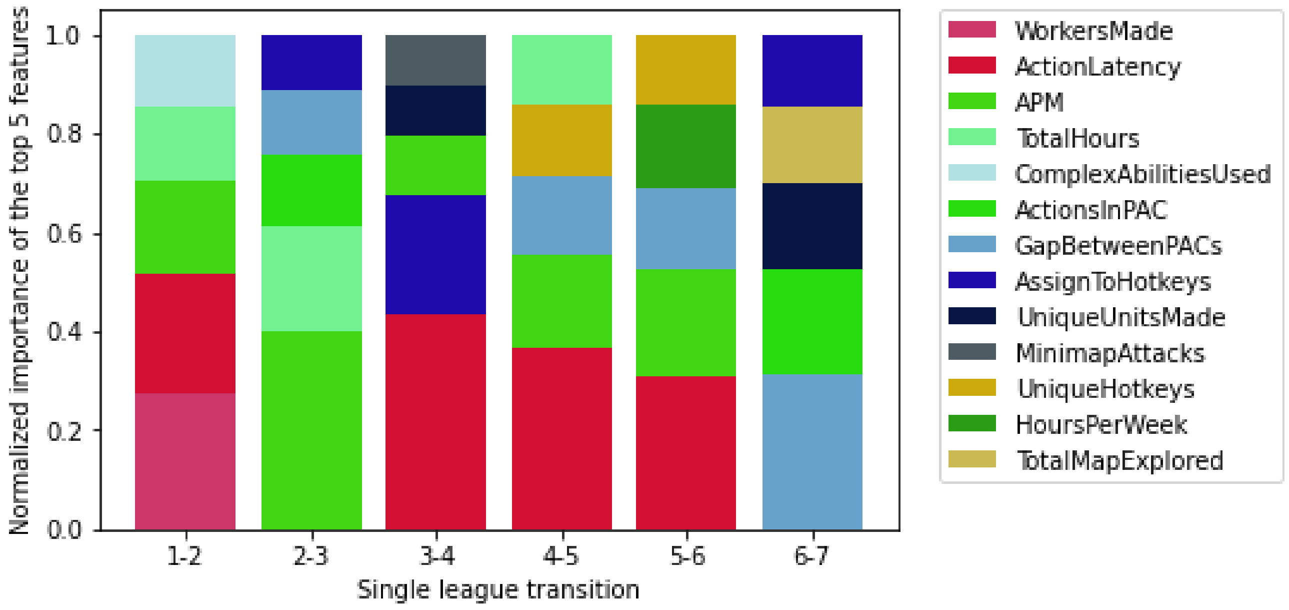

Figure 9. The top 5 feature importances for each league transition by Logistic Regression

We see very similar feature importances by league transition in *Figure 9* as we do *Figure 7* which is heartening for our shap detection in *Figure 8* as Action Latency appears to have high importance for transitions from league 3 to 6. We can pretty conclusively say that once you advance beyond the basics and make it into league 3, one of the most effective ways to increase your league index is to focus on reducing your Action Latency.

Anti-Smurf Detection

In Starcraft, Smurfing generally creates a less than enjoyable play experience for those people properly ranked in the lower ranks as they play against players whose skill level is beyond theirs. For this project, we will assume that Blizzard, the company that owns Starcraft and supports online play of the game, thinks smurfing is very bad, and as such the penalty for smurfing is getting your secondary account, from here on out referred to as the smurf or smurf account, banned. We will attempt to use predictive machine learning trained on our data set to take the input stats from a game played by a specific player and produce whether or not they are in the low leagues, which we define as leagues 1 to 3, or high leagues, which we define as leagues 4 to 7. We tested three machine learning algorithms, logistic regression, random forest, and support vector machines (SVM). After optimizing for hyperparameters, we found that logistic regression and SVM both outperform random forest. *Figure 10* shows us that Logistic Regression is very slightly

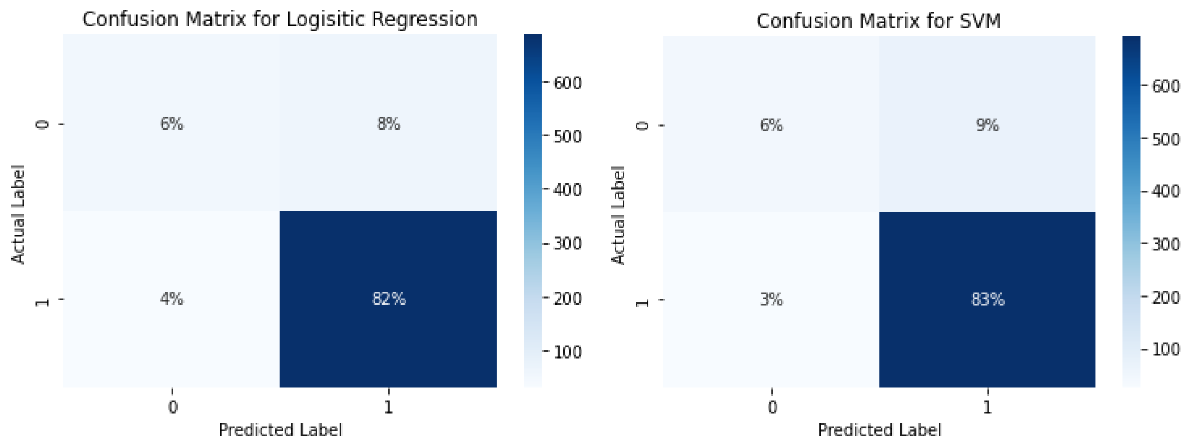

Figure 10. Confusion Matrices for Logistic Regression and SVM. Label 0 is Low League, Label 1 is High League

less likely to incorrectly predict someone as smurfing versus SVM. We know that, given the severe punishment our hypothetical Blizzard is enacting on smurfers, our model needs to be most concerned with keeping the false positive rate small while not falling into the trap of simply letting everyone through as we would rather miss some smurfers than make a mistaken ban, yet we do still want out model to have some effect so we move forward with Logistic Regression. *Figure 11* gives us some insight into what might be a potentially useful threshold to select as we will value precision over recall as precision is larger when the false positive rate is small.

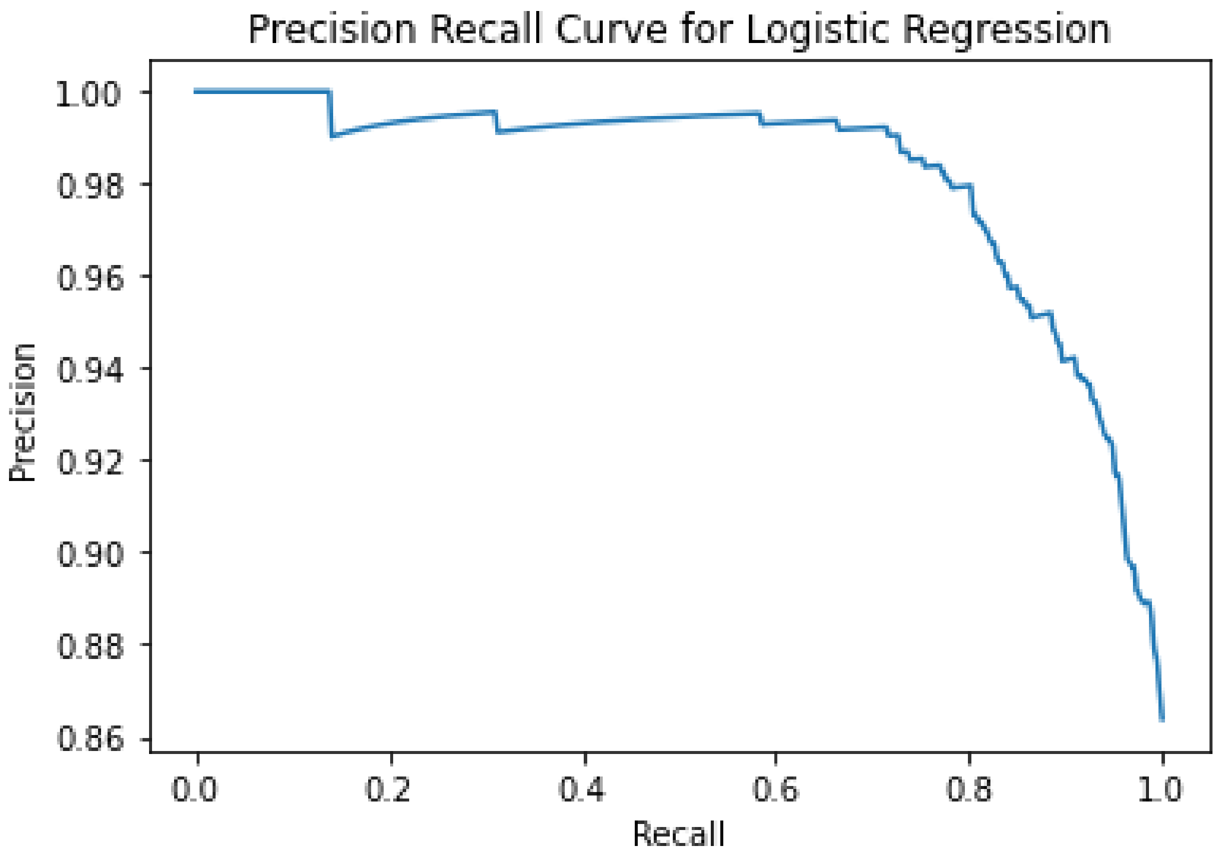

Figure 11. Precision-Recall curve for Logistic Regression

As stated above, our specific business problem values precision over recall. We are interested in picking a threshold which strikes a balance between a large value of precision without allowing recall to plummet. We can see from *Figure 11* that there appears to be a point at about a precision of .98 and a recall of .8. We plot each of precision and recall as well as our F0.5 score against the thresholds for a clearer picture.

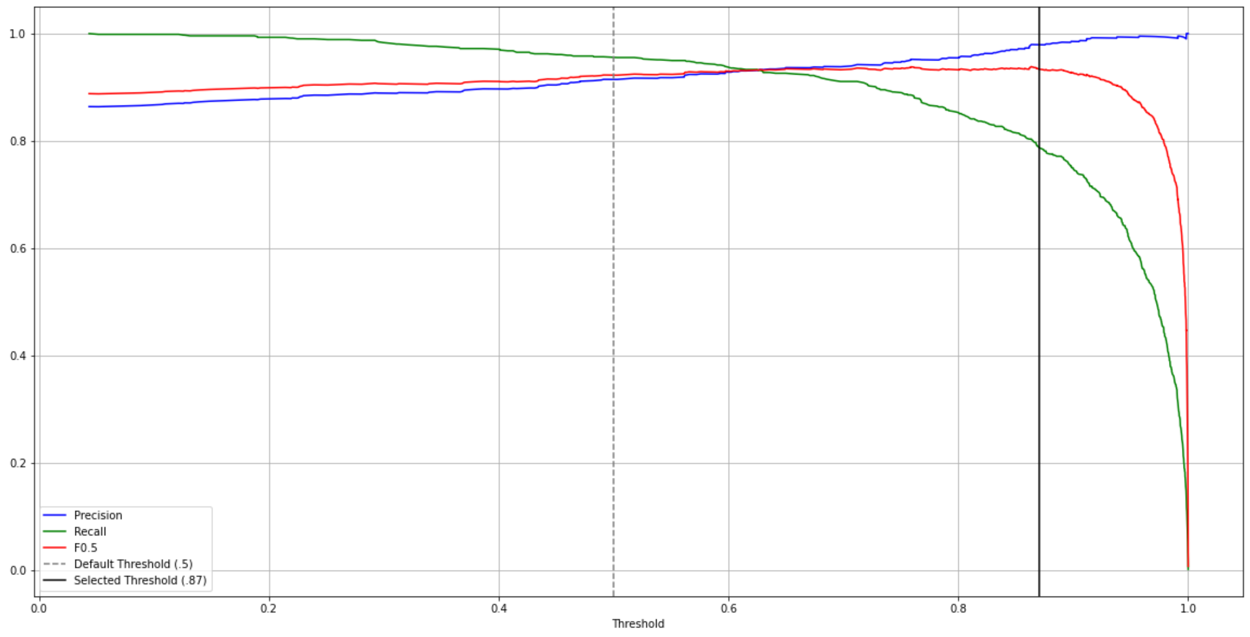

Figure 12. Precision, Recall, and F0.5 plotted versus the thresholds

We can see from *Figure 12* more clearly that as precision increases, recall decreases but that our F0.5 score continues to increase. We noted a precision of .98 and recall of .8 above in *Figure 11* and we can see that our chosen threshold of .87 seems to correspond to that exact point. It is worth noting that this point is about as far right along the thresholding axis in *Figure 12* we can go before we start to see our F0.5 score drop. Thresholded at .87, our model will still detect some Smurfers while erring on the side of letting the players play so as not to misclassify a non-smurfer and lead to an unjustified ban. We rerun our predictions with this new threshold and see how our new model parameters classify our test set.

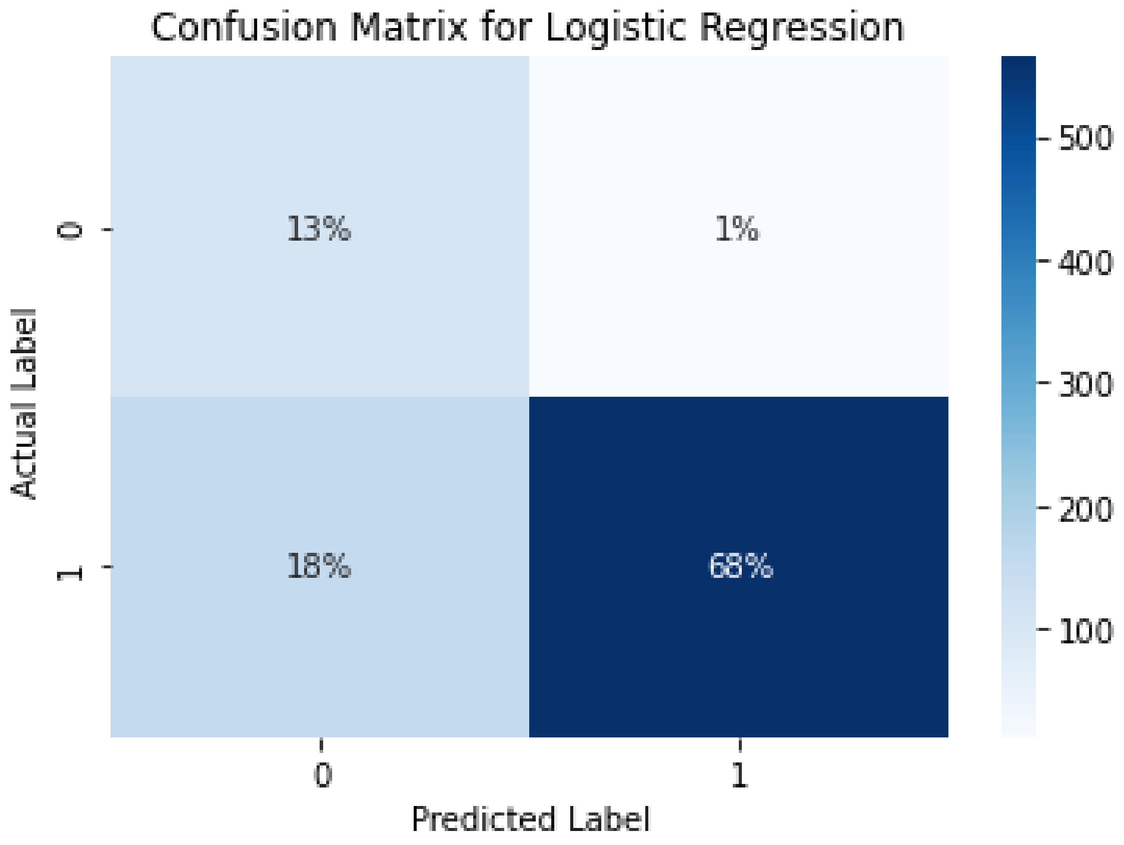

Figure 13. Confusion Matrix for Logistic Regression after thresholding at .87

Figure 13 shows us our new confusion matrix. We've successfully reduced those players who are actually low league getting predicted as high league to a low percentage. This reduction will help prevent us incorrectly labeling a player as a smurf when in reality they are not. We also notice that the number of high league players who are predicted as low league has increased. This is where some number of our smurfers who we miss will sit, but this is acceptable as we would much rather miss smurfers than ban a non-smurfer for smurfing. We examined which rows of our test set were mislabeled manually and found that, generally, the few players who were actually low league but predicted as high league tended to have a higher APM and lower Action Latency than the average APM and Action Latency for correctly predicted low league players. Given that league index is directly related to how often you win, this suggests that these players may have a strategic fault rather than a mechanical one, which is nearly impossible for our model to take into account. We found that those players incorrectly predicted as low league when they were high league frequently had a combination of low total hours, low APM, and high Action Latency when compared to the average high league player. Notably, players who had low total hours but high APM and low Action Latency are generally still classified as high league by our model. Given that this is expected makeup of a smurf, it is heartening to see this prediction.

	Precision	Recall	F1 Score	Support
Low League	.41	.90	.56	118
High League	.98	.79	.87	717
Average/Total	.90	.80	.83	835

Figure 14. Classification Report for Thresholded Modeling

As expected from *Figure 12*, *Figure 14* shows us that we sacrifice the low league precision in order to increase high league precision, which we care about as mislabeling someone as high league leads to incorrect smurf detection. Our overall precision is also relatively high without sacrificing recall to the point of making our model unusable.

Conclusion

We examined a large amount of data relating to Starcraft II players. We broke down each league transition and found which skills were most likely to cause you to increase your league index by one. Of note, we found that maintaining a low Action Latency frequently plays a large role in increasing league index, specifically in the middle leagues, where the majority of the player base tends to clump. We also found that until the absolute highest of leagues, total hours spent on the game are a pretty solid indicator of success, and that if you are stuck in a lower league, simply playing more tends to be an effective way of increasing your league index. We also simulated “Smurf” detection, looking for players creating new accounts to play at low leagues disproportionate to their skill level. We thresholded our predictive model to protect against incorrect smurf labeling and found a model that tends to catch players with low total hours who otherwise play at a high league level.

For future work implementing a live stream of data rather than a static input may result in more useful smurf detection as their is marked difference between a player playing above their skill level for a small period of time, colloquially known as being “in the zone”, and a player consistently playing above the average skill level at their league index, which is more likely to be smurfing. Rather than averaging these statistics across a single game, taking a real time look at these statistics would allow for more precise modeling.

References

1. UCI Machine Learning Repository: Skillcraft1 master table dataset data set, https://archive.ics.uci.edu/ml/datasets/skillcraft1+master+table+dataset Accessed December 8, 2021