Anisah Alahmed

Predicting the Outcome of Dota 2 Matches Based on Draft Picks

Introduction

This was an attempt at training a model to produce probabilities for the binary outcome of matches of the strategy game Dota 2 based solely on the initial conditions of the game, known as the “draft”. Dota 2 as a strategy game is quite complex and as a result predicting the outcome becomes more difficult the less information you have about the game, and the beginning of the game is when the least information is available to an observer, as advantages appear over time. However, there is a clear strategic emphasis on a team entering the game with a favorable draft, and thus that information can be applied to predict a slight favor in these initial conditions. The goal for the models trained is to learn these slight advantages and determine who starts with the upper hand, without any information about the players themselves.

The game cycle of Dota 2 involves two teams, called “Radiant” and “Dire”, of five players who control individual characters known as “Heroes” with the ultimate goal being to destroy the “Ancient” belonging to the enemy team while protecting their own. The first team to destroy the Ancient of the other wins the game. Each player controls their own character and only their own character, meaning autonomy of each player is high, while also making team cooperation imperative for success. This cooperation starts with the selection of the Heros at the beginning of the game, as each one has different attributes and abilities that manifest advantages and disadvantages when it comes to the matchups against the enemy team as well as the synergies on one’s own team.

Data

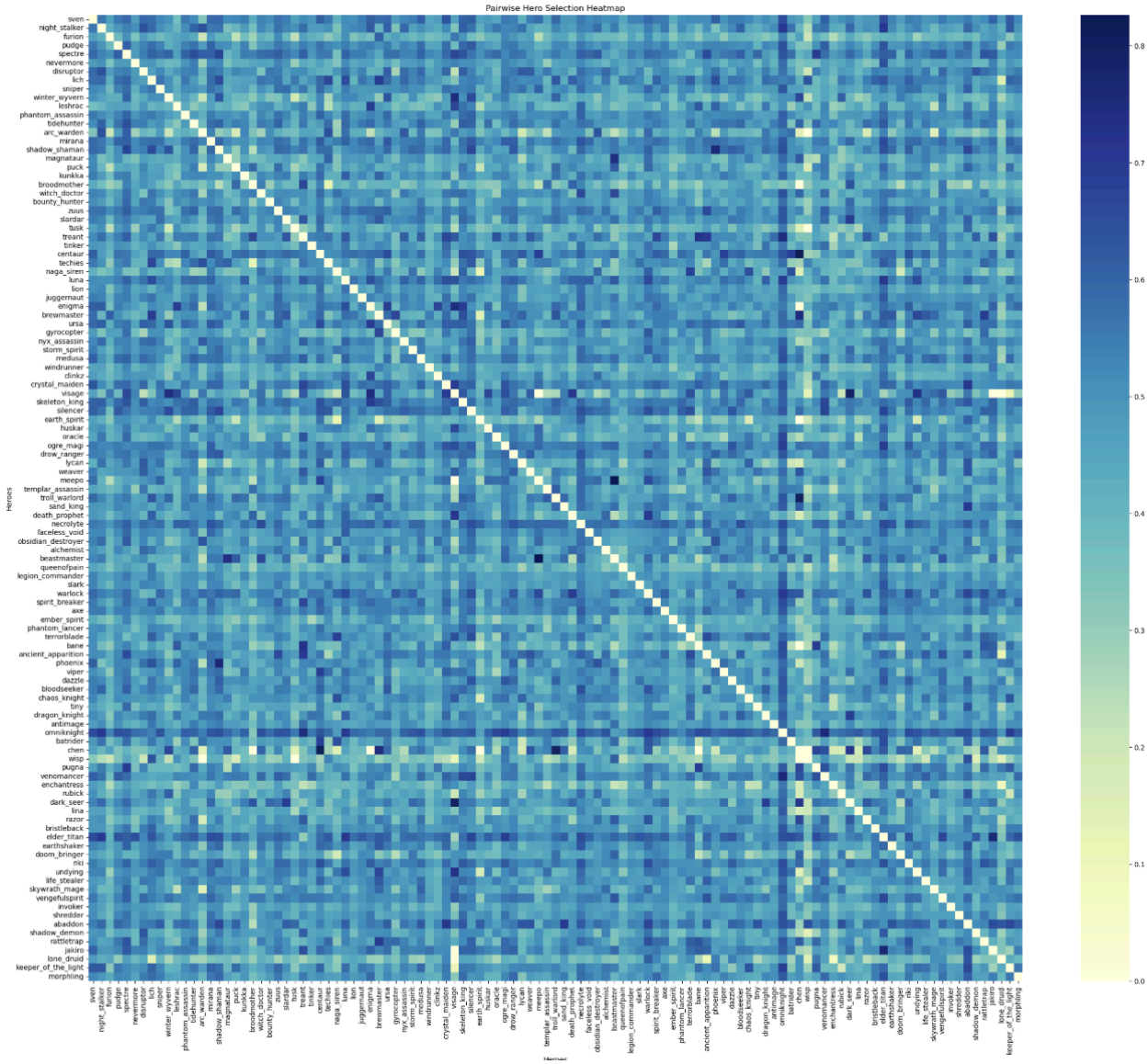
The dataset, “Dota2 Games Results” (Tridgell, Stephen.), contains information about the initial conditions and the outcomes of ~100,000 games of Dota 2 from 2017. The information provided about each game is the mode of the game, which informs how the draft was selected by the players, the competitive nature of the match, which informs how motivated players may be to perform well, and 113 columns with indicate whether or not a given character, called a Hero was selected and for which team that hero was picked. A hero can only be picked by one team. A hero selected for the Radiant will have its value set to “1” for that match, while a hero select for the Dire will be “-1”. The

game requires exactly 5 heroes for each team, which means each row in the dataset will have exactly 5 1-values and 5 -1-values. The value for every other hero in the game which was not selected will be zero. The dataset also provides the outcome of each match with a similar representation scheme: 1 for Radiant victory, and -1 for Dire victory.

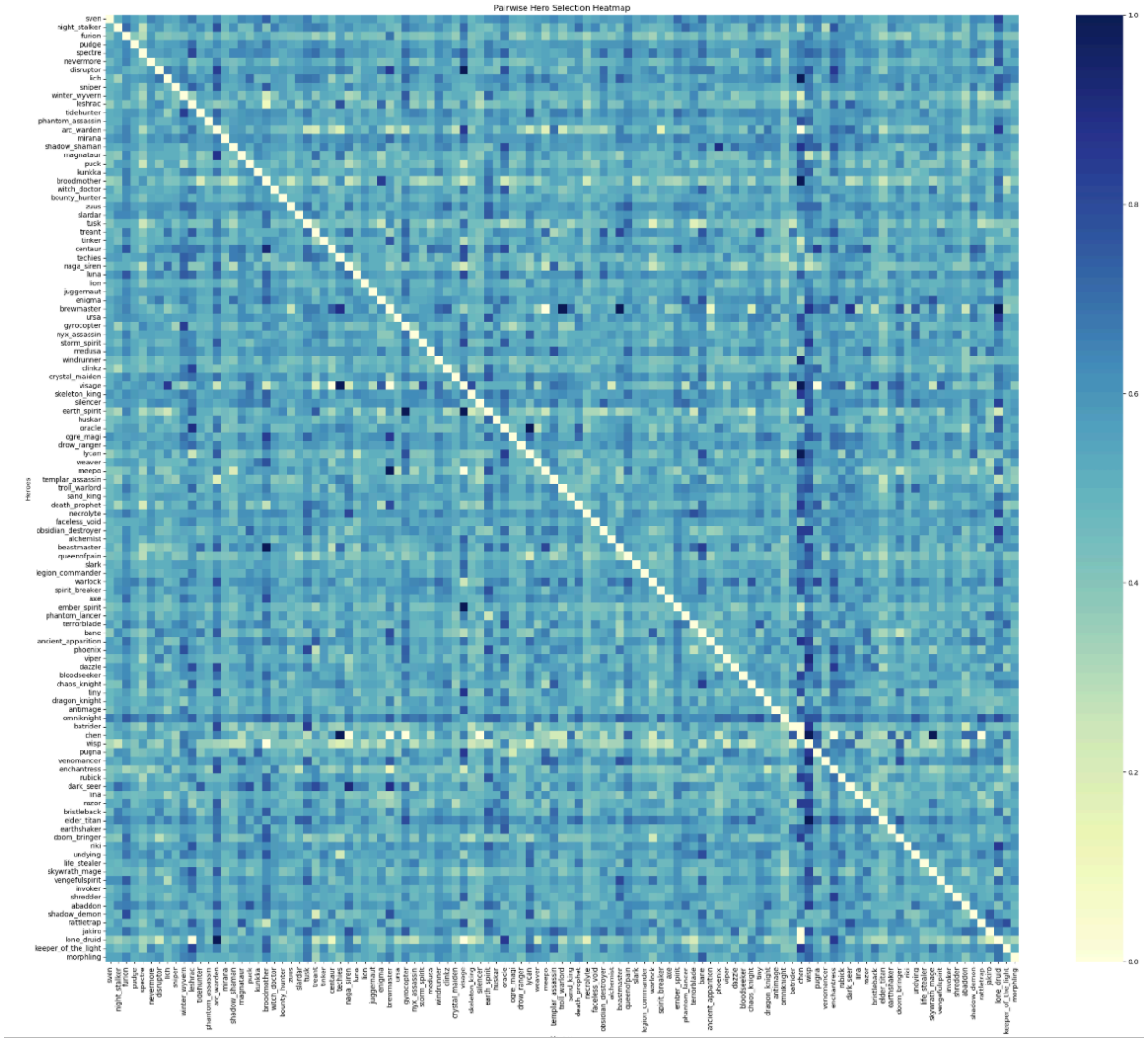
Visualization

With the information available in this dataset, the latent features that the models will be intended to learn will be the ways which different sets of heroes on the same team are able to boost each other, as well as the ways in which different pairs of heroes on opposite teams can pose challenges to the players.

When it comes to the viability of a selected hero and the contribution of that hero to the probabilistic success of the team, the space of outcomes would require 10 dimensions to represent. The impact of any given hero based on the draft is a delicate balance between how all 4 of that hero's allies help and how all 5 of that hero's enemies hinder. However, within regular play of Dota 2, it is often the case that the selection of any hero by a player is made with a specific synergy or counter in mind. These would be simple 2-dimensional relationships and should be much easier emergent features to be learned, and we can also represent these pairs with heatmaps.



This heatmap represents the success rate of pairs of heroes on the same team relative to how many times that pair appears in the dataset. Worth noting is the fact that this is symmetrical because pairs of heroes on the same team share the same victories. This is in contrast to the alternative heatmap for opposing matchup pairs:



The win rate values on this chart are for the heroes on the vertical access when paired against heroes on the horizontal axis. In this heatmap we can initially see that it is not symmetrical, and this is because these values miss the additional context of the complete team composition of both sides. Additionally, because these pairs to do not share “victories” for the purposes of the calculation, the win rate is not a straightforward flip due to the varied impacts of different team dynamics. However, we can clearly make out two thick vertical lines over the heroes “Chen” and “Wisp”, which means that these heroes generally have poor win rates overall.

These visualizations highlight some basic relationships that are present in the data that ideally we should be able to coerce a model into recognizing.

Preprocessing

The preprocessing necessary for using the data as is was minimal. For the models selected here, the ‘gamemode’ and ‘gametype’ columns were dropped from the feature set. This is because these do not bear much weight on the outcome of the match directly.

Additionally, for the purposes of working well with the built-in algorithms for the `scikit-learn` library, the target values were converted from a binary representation of `1` and `-1` to `1` and `0` for Radiant and Dire victories respectively.

Fortunately, because all of the features in this dataset are discrete, and the selection representation happens to be centered around 0, no scaling is required. Additionally, the `scikit-learn` implementations of the algorithms used add bias vectors internally when necessary, meaning that step can be removed from the preparation.

Finally, the dataset was split into training, validation, and testing sets at an 80/16/4 split.

Methods

Model 1: Logistic Regression

A logistic regression model was chosen for the following reasons:

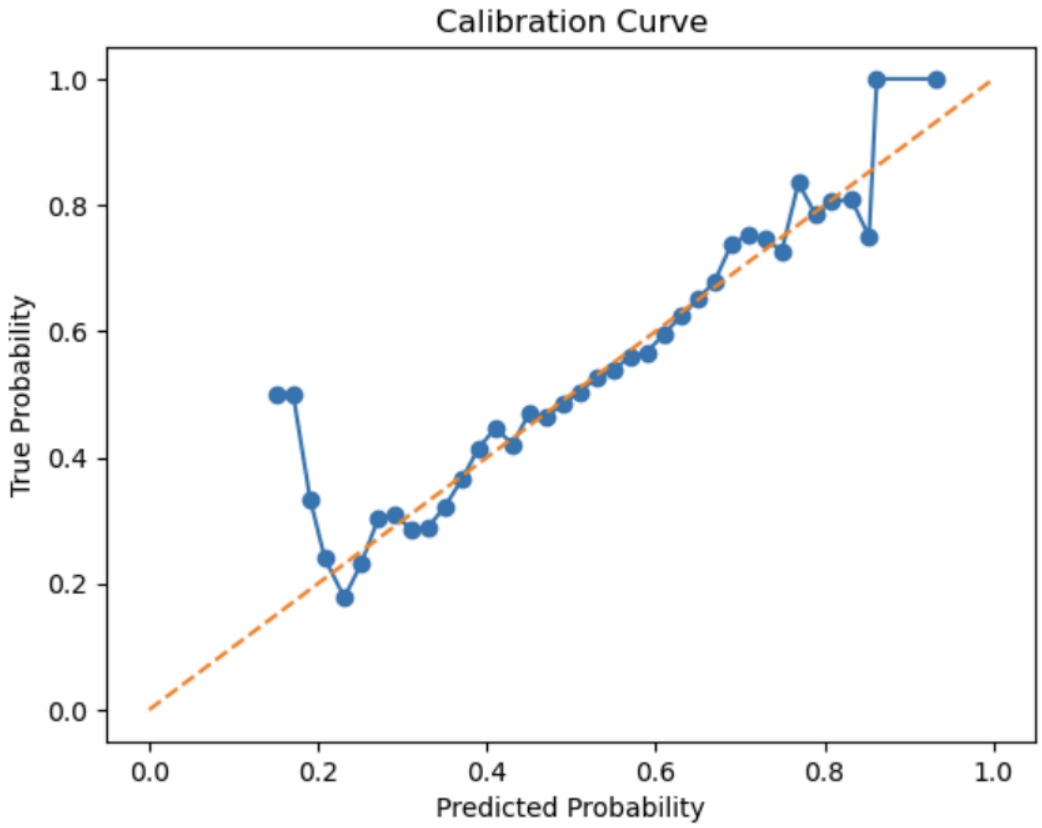
- Target is technically binary classification
- Gives probabilities which is what we want
- The impact of different draft selections is a linear transformation

After some experimentation, calibration fitting was applied to the regression model, where a sigmoidal method seemed to have the best results. Different regularization methods were also applied, to negligible effect.

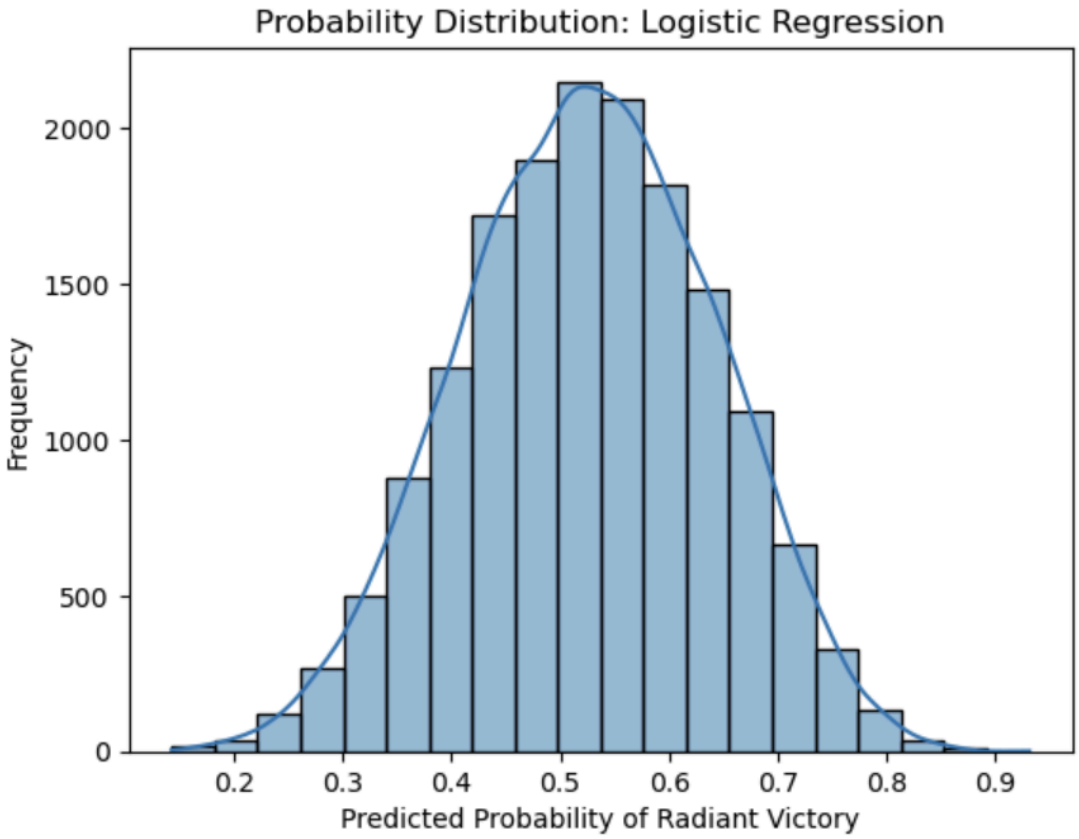
The model was set up using the scikit-learn built-in `LogisticRegression` and `CalibratedClassifierCV` utilities.

Results

The regression model achieved a 60% accuracy on the training data and a 59% accuracy on the validation set. This is actually comparable to how an experienced player might fare with this amount of information, however a 60% might sound quite unnerving. To figure out whether or not the 60% rate is a “meaningful” prediction, we can analyze the calibration statistics for the model with both a Briar score and a Calibration Curve.



As the curve shows, the model makes mostly reasonable predictions for the majority of the samples. However, for samples where an extreme prediction for probability (80% or more in either outcome) would have been warranted, the model chose more conservative predictions. We can probably get an idea why by looking at the spread of predictions:



We can see that the predictions have a smooth normal distribution. In general, the model made very few predictions for those poorly calibrated extremes. I theorize that the model learned is that it would be easier to get the correct answer something by clamping the distribution closer to 50% than picking some of the relationships we visualized above out of the noisy dataset. Still, using the `brier_score_loss` utility from scikit-learn the model achieves a score of 0.2357 (better than a 50/50 toss-up score of 0.25), which in combination with the calibration curve supports the “meaningfulness” of the low-confidence predictions made by the model.

Model 2: Random Forest

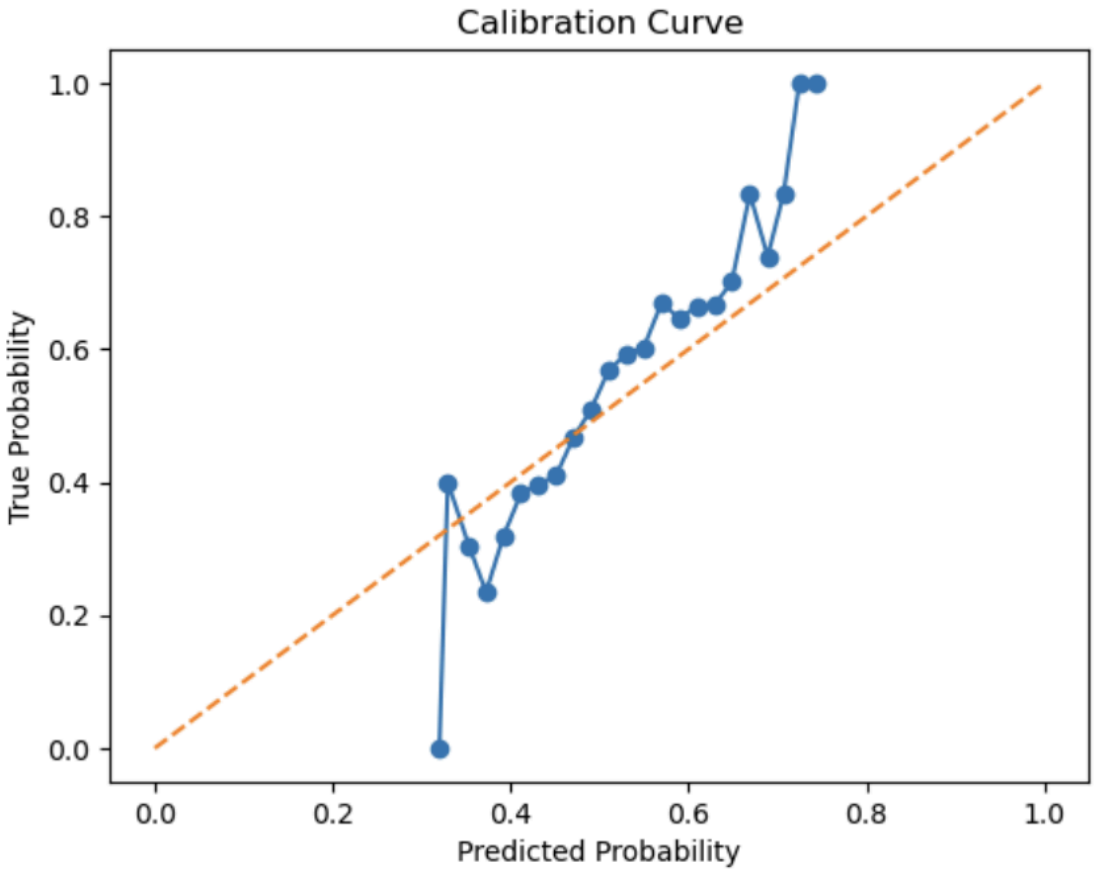
This algorithm was chosen for the following reasons:

- Resistant to overfitting (in theory, see results)
    - https://stats.stackexchange.com/questions/36165/does-the-optimal-number-of-trees-in-a-random-forest-depend-on-the-number-of-pred
- Internally prioritizes features
    - The thinking is that it would have an easier time picking out stronger heroes and pairings

Experimentation showed poor results, so in an effort to get optimal validation results, the `GridSearchCV` utility was used, testing out many different hyperparameters for the model, including tree count, tree depth, and branching thresholds.

Results

This model suffered from severe overfitting, despite many different hyperparameter combinations, regularization techniques, and calibration being applied. The calibration for the final attempt appears as below:



With a training accuracy of 74% but a validation accuracy of 58%. What I have come to understand is that despite the theoretical resistance of random forest to overfitting generally, this attribute does not hold against exceptionally noisy datasets (O_Devinyak et al.).

Because our dataset is composed of 113 features, but only 10 features are actually meaningful for any given sample, our dataset definitely meets the definition of noisy.

Conclusions

Given the results here, my conclusions are that logistic regression is a suitable model for partially informing predictions, though its underconfidence and its inability to predict extreme results makes it unreliable as a general purpose prediction tool. Random forest

seems to be completely unsuited to this problem, at least without significant additional engineering of the features, as its sparseness and high noisiness break the natural fitting resistances of the algorithm.

Theories and Potential Improvements

- When looking at the prediction distribution for the logistic regression, it follows a very standard normal distribution. It is possible that this is a projection of the distribution of the input data. This could be investigated, and if it is found to hold, then perhaps Gaussian Naive Bayes would be a more suitable aggregating model for this problem.
- The selection of the random tree model was done in place of a more opaque neural network, but perhaps such a network with multiple layers would be able to resist the noise of the dataset as well as learn the latent features in the data regarding the different combinations of heroes and how they interact with each other,
- Those latent features could in theory be extracted manually and used to inform more sophisticated models. Those relationships were already calculated for the purpose of visualizing the synergies and the counters in the heatmaps above. Perhaps factoring that knowledge, with room for more than just two dimensions, would result in a general improvement of fitting performance, and may allow for poorly performing models like random forest to be more efficiently informed and enabled to leverage their strengths, instead of relying on the models themselves to learn those patterns for us.

References

Tridgell, Stephen. "Dota2 Games Results." UCI Machine Learning Repository, 2016, https://archive.ics.uci.edu/dataset/367/dota2+games+results.

O_Devinyak et al. “Does the Optimal Number of Trees in a Random Forest Depend on the Number of Predictors?” Cross Validated, 12 Sept. 2012, stats.stackexchange.com/questions/36165/does-the-optimal-number-of-trees-in-a-random-forest-depend-on-the-number-of-pred.

Acknowledgements

ChatGPT was used for the following resources:

- Exploring new machine learning algorithms and tools not learned in class (e.g. random forest, calibration)
- Explaining built-in scikit-learn utilities
- Generating some of the code in the notebook, primarily regarding graphics

GitHub
https://github.com/anisahalahmed/ITCS3156-final/blob/main/Untitled1.ipynb