

Scrabble Player Rating Competition

Syed Raza, Gaurav Kumar, Kashi Bondugula, Venkata Nagabhairu

1) Abstract

In this report, we summarize the results of applying several regression algorithms to a tabular dataset consisting of over seventy-three thousand recorded games of online Scrabble, taking place between human players and automated bots on the popular website, Woogles.io. The goal was to predict the value of one of the attributes that each player account on Woogles.io has, their rating. We compare the results of 2 non-parametric and 2 parametric supervised learning algorithms for this purpose, as well as that of an ensemble. We divide our dataset using an approximate 66/34 split, and ultimately achieve a mean RMSE of 2.5 on the test portion of the dataset. We finish by mentioning shortcomings of this work, as well as potential improvements going forward.

As of December 13th, our team ranked 153rd place in the Kaggle competition.

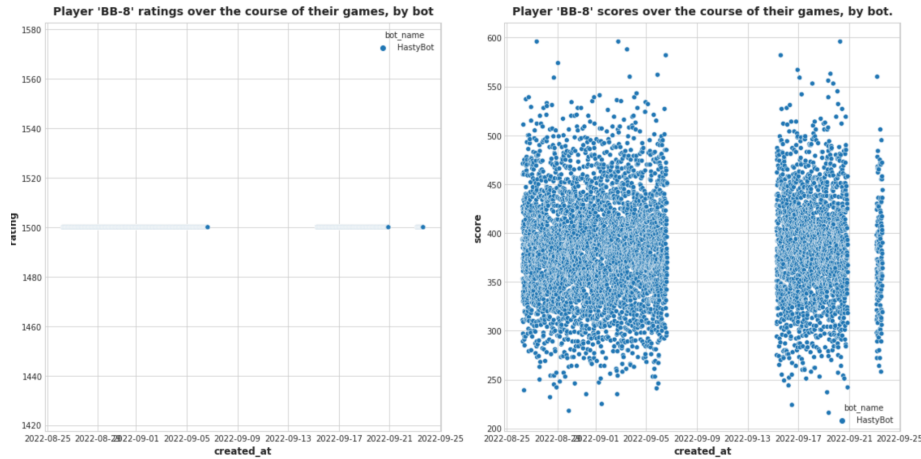
With regards to the portion of work - we collaborated equally on the exploratory data analysis, preprocessing, modeling and evaluation sections of this project. The modeling section (i.e. section 4) of this report was divided up equally between us. The captain, Syed Raza, wrote the other sections.

2) Background and Exploratory Data Analysis

In this competition, we performed an analysis and supervised regression on a tabular dataset from Kaggle. This dataset was donated by Woogles.io, a popular online platform where players play Scrabble against automated “bots” who have been programmed to be their opponent. Specifically, this dataset contains records related to roughly 73,000 Scrabble games played by humans versus three bots on Woogles.io: BetterBot (beginner), STEEBot (intermediate), and 3) HastyBot (advanced).

Our regression task was, given a record of a game on Woogles.io against one of the three aforementioned bots, to predict the “rating” of the human players before that game started.

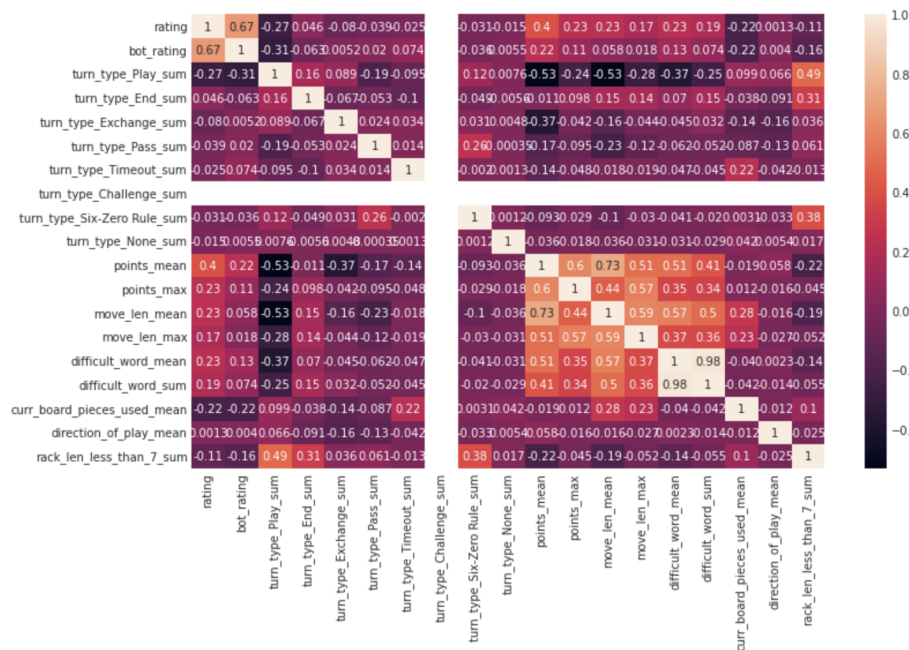
Our analysis [1]–[3] uncovered several observations that informed our modeling approach. For example, we found there are some accounts, such as “BB-8”, that only ever have a 1500 rating. **Figure 2.1** displays the scatter plots of ratings and scores of this player for all their games:



We consider player accounts like “BB-8” to be anomalies in the dataset. We decided to exclude them from our modeling, since we are not given any of the ratings for any human player in the test set; therefore, we assume none of them are these anomalous players.

3) Feature Selection and Preprocessing

To create a more holistic view of each game, we summarize and join multiple columns across the CSVs in the provided dataset, to create a single table that displays aggregated metrics for each player’s game. This allows to create a correlation matrix, visualized in **Figure 3.1** below, that shows how strongly different features are interrelated:



Based on this heatmap, we narrowed down the features to use for our model to just 5 categorical features, in addition to all the features representing metrics for each game (e.g. “difficult_word_sum”, or “max_overtime_minutes”). For preprocessing, we lastly standardized the numerical columns using Z-scores, and encoded categorical columns using one-hot vectors.

4) Modeling and Evaluation

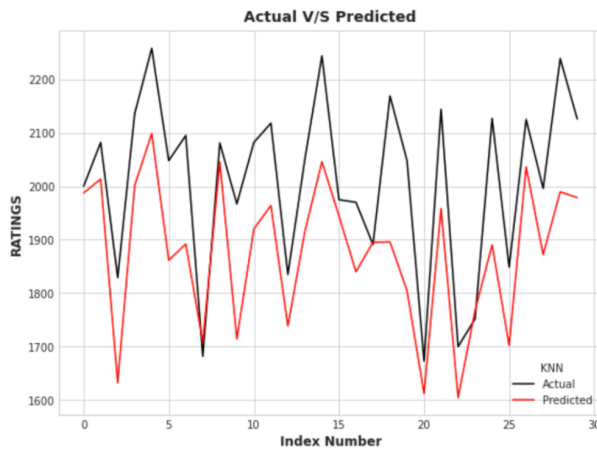
We begin this section by broadly surveying several parametric and non-parametric approaches [4]–[6] for regression analysis. **Figure 4.1** shows a table of these initial models [7], each with their RMSE scores on the train and test set, and the difference in between thereof:

	model_type	Training	Validation	train-test-skew
0	Nearest Neighbors	112.661804	138.789492	26.127688
1	Linear SVM	148.293478	148.395279	0.101801
2	Decision Tree	144.847661	146.357653	1.509991
3	Neural Net	114.441591	121.189335	6.747744
4	Elastic Net	152.242892	152.337510	0.094618

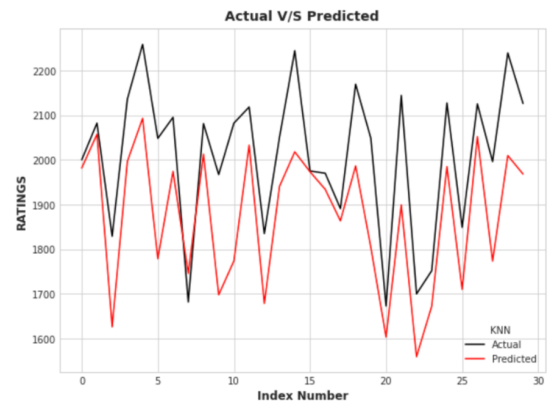
4-1 Non parametric Models

K-nearest neighbors (KNN) regression is an instance-based learning method. It works by storing all available cases and predicting the numerical target based on a similarity measure (distance functions). In KNN regression, the output is the average or median of the values of its K nearest neighbors. obtained an RMSE value of about 153.019. **Figure 4.2** visualizes the change in RMSE before and after grid search.

Without Hyperparameter Tuning (RMSE : 162.08)



After Hyperparameter Tuning (RMSE : 162.01)



The Decision Tree regressor breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. Applied the different metrics and keep maxdepth as 10 and at that depth we get a somewhat good test RMSE value of about 183.2.

4-2 Parametric Models

Based on the support vector machines method, the Linear SVR is an algorithm to solve the regression problems. The Linear SVR algorithm applies a linear kernel method and it works well with large datasets. The L1 or L2 method can be specified as a loss function in this model. Use c value as 1.0 and fit to our model. We get RMSE value as 170.5965.

Neural network (NN) using Scikit-learn multilayer perceptron regressor has been used as the model. Neural networks are known to be capable of recognizing non-linear relations well in large datasets, which is the case with the given dataset. As NN is a parametric machine learning algorithm, using different hyper parameters the models result in different accuracies. Lesser iteration never converges. For the given data set hyper parameters [5] with: { max_iter=1500, alpha=1, learning_rate='adaptive' } worked best. Although the model was overfitted for different alpha (to penalize the weights), iterations and other hyperparameters values. To overcome that, the model can be enhanced by adding multiple hidden layers and trying dropout neurons for regularization[10]. The below graph (Fig 4.3a) shows the loss curve as the iterations increase [8]. Figure 4.3b shows how the model RMSE value changes for different alpha, yet it has an overfitted result when observing the train-test skew.

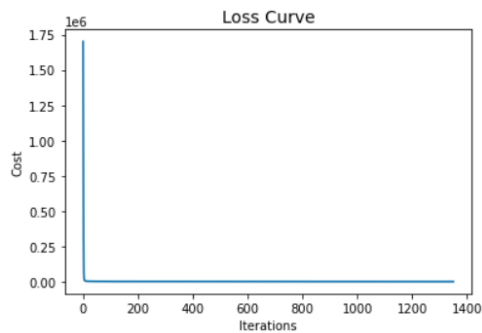


Fig 4.3a

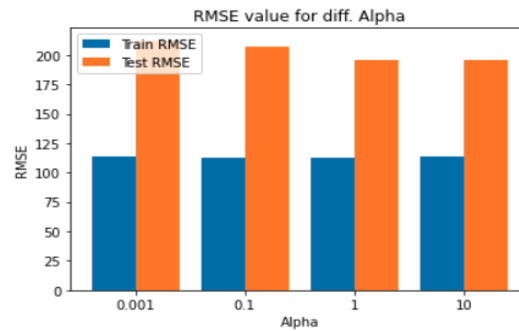


Fig 4.3b

Kaggle submission for the above model gave a result of 156.17987

4-3 Stacking Model



For our ensemble model, we stack a linear model with l2 regularization as the meta-regressor [9]. We use l2 over l1 because we have already performed feature selection ourselves above utilizing the heatmap in section 3. We stack this ridge regressor on top of cross validated base learners (one is a linear regression model optimized using stochastic gradient descent, and the other is a decision tree) to reduce the chances of overfitting. This model suffered from an absurdly high train-test skew - it achieved an average RMSE of 133.5 on the train data, while on test data its average RMSE was about 2.5.

5) Conclusions and Future Work

In this project, we have conducted both an analysis and supervised regression on a tabular dataset. As you can see, using a heatmap and other visualizations we were able to narrow down our set of predictors to just a few columns of categorical variables, as well as numerical values pertaining to aggregate metrics regarding each Scrabble game. After attempting several individual models, we arrived at a stacking model that achieved a mean RMSE of approximately 2.4 on the test portion of the dataset.

One of the major challenges we faced in this project was the difficulty in reproducing each other's results in our Kaggle notebooks. If given more time and resources, we would further investigate this issue by testing our code on different cloud-provided Jupyter kernels, and perhaps even on our local machines. Additionally, we would likely address the issues in overfitting by re-examining the amount and kinds of feature engineering we did, so that we could attain a model with both lower complexity and lower train-test skew.

Works Cited

- [1] “textstat/easy_words.txt at master · textstat/textstat.”
https://github.com/textstat/textstat/blob/master/textstat/resources/en/easy_words.txt (accessed Dec. 09, 2022).
- [2] “SCRABBLE Rules | How To Play Scrabble | Rules of Scrabble EXPLAINED - YouTube.”
<https://www.youtube.com/watch?v=IG1QAYWvKIQ> (accessed Dec. 09, 2022).
- [3] “ Scrabble, EDA+FE+Modeling .<https://kaggle.com/code/hasanbasriakcay/scrabble-eda-fe-modeling> (accessed Dec. 09, 2022).
- [4] “sklearn.tree.DecisionTreeRegressor,” *scikit-learn*.
<https://scikit-learn/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html> (accessed Dec. 09, 2022).
- [5] “sklearn.neural_network.MLPRegressor,” *scikit-learn*.
https://scikit-learn/stable/modules/generated/sklearn.neural_network.MLPRegressor.html (accessed Dec. 09, 2022).
- [6] “sklearn.svm.SVR,” *scikit-learn*. <https://scikit-learn/stable/modules/generated/sklearn.svm.SVR.html> (accessed Dec. 09, 2022).
- [7] “Classifier comparison,” *scikit-learn*.
https://scikit-learn/stable/auto_examples/classification/plot_classifier_comparison.html (accessed Dec. 09, 2022).
- [8] “NN - Multi-layer Perceptron Regressor (MLPRegressor) - Michael Fuchs Python.”
<https://michael-fuchs-python.netlify.app/2021/02/10/nn-multi-layer-perceptron-regressor-mlpregressor/> (accessed Dec. 12, 2022).
- [9] “sklearn.ensemble.StackingRegressor,” *scikit-learn*.
<https://scikit-learn/stable/modules/generated/sklearn.ensemble.StackingRegressor.html> (accessed Dec. 09, 2022).
- [10] “Regularization in a Neural Network | Dealing with overfitting”, AssemblyAI
<https://www.youtube.com/watch?v=EhRcPoIM-Q> (accessed Dec. 19, 2022).