Prediction of Scrabble Player Ratings

Using data from Woogles.io

Spencer Stromback, Jessica Cheng, Cindy Wang, Cody Ortloff, Yumi Yu

Project Context & Definition

Woogles.io is a gameplay website that hosts scrabble games, and it offers the choice of playing against 3 different levels of robots for the players. The primary goal of this project is to develop a predictive model that can accurately estimate the ratings of human players based on their gameplay in Scrabble matches. This estimation is based on comprehensive gameplay data collected from approximately 73,000 matches played between human players and three different bots (BetterBot, STEEBot, and HastyBot), each representing varying skill levels from beginner to advanced.

What is Scrabble?:
Scrabble is a board game where up to four players take turns placing letter tiles on a board to form words. The amount of points each player receives is determined by the letters they play on the board, and where those letters are played. The player with the most points at the end of the game wins.

The tiles are drawn from a bag by each player at random. There are 100 tiles in total. Each tile has a single letter, along with a point value associated with that letter. Common letters, such as vowels, have lower point values, while less common letters, like Q and Z, have higher point values.

After a word is placed, the player randomly replaces the same number of tiles that were used until they once again have 7 tiles.

Blank Tiles can represent any letter, but do not gain the player any points. Also, the distribution of available tiles is not equal among all letters. Some letters are far more common than others. For example, there are 12 E's in a Scrabble bag, while there is only one Q.

The board is a 15 x 15 square grid. Each space on the grid holds a single letter tile. Some spaces have special score multipliers associated with them. Double letter score and triple letter score spaces multiply the points gained for whatever letter is placed on that space. Similarly, there are also double and triple word score spaces, that multiply the score gained for an entire word that passes through that space.

There are 5 primary types of turns in a game of scrabble:

1. Place a word: The player places whatever letters they like to form a word on the board. Their word must be connected to an existing word on the board (unless it's the first word placed)
2. Exchange: If the player does not like their tiles, they may elect to spend their turn exchanging some or all of their tiles for new ones.
3. Pass: A player may elect to pass their turn. While uncommon, this can come up if a player has good tiles but cannot think of a word to place.
4. Challenge: If a player thinks that a word placed by another player is not a real word, they may challenge the play. If the challenge is successful, the tiles are removed and the placing player loses their turn. Whether or not there is a penalty for an unsuccessful challenge is dependent on the rules of that specific game.
5. Bingo: If a player makes a word placement that uses all seven of their letters, this is called a "Bingo". This results in an additional 50 points for that play.

The game ends when one of two conditions is met:

1. All tiles have been drawn and one player runs out of tiles.
2. All players pass their turns in succession

The player with the highest final score wins.

Dataset Description

The dataset encompasses multiple components, each offering insights into different aspects of the games:

1. Games Metadata (games.csv): This file provides essential metadata for each Scrabble game, including unique game IDs, player information, time controls, game outcomes, and additional game settings like lexicon used and time limits.

2. Gameplay Data (turns.csv): Detailed turn-by-turn data is available in this file, documenting each move's specifics, including player racks, move locations, points scored, and turn types.

3. Player Scores and Ratings (train.csv and test.csv): These files are crucial for the model's training and evaluation. 'train.csv' comprises players' final scores and their ratings before the game, used for training the model. 'test.csv' includes similar data but lacks the players' pre-game ratings, which the model aims to predict.

Solution Overview

In this project report, we will outline our systematic approach to solving this challenge step by step. We started with a very comprehensive data preprocessing and feature engineering process to generate and transform the raw features into more informative metrics. With visualizations of the player statistics, we were able to further understand and clean up our data to prepare for modeling. After aggregating the features at hand, we proceeded to build Random Forest, XGBoost, and LightGBM models to predict the players' ratings at best.

Data Preprocessing and Feature Engineering

We began with an intensive data preprocessing stage, where the raw data was meticulously cleaned and transformed. Our focus was on:

- Cleaning and normalizing data to ensure consistency.
- Engineering new features that could better represent player skills and strategies.

Data Visualization and Analysis

To deepen our understanding of the dataset, we employed various data visualization techniques. These visualizations provided insightful views into player statistics and gameplay patterns, enabling us to:

- Identify trends and outliers in the data.
- Further refine and clean the dataset, ensuring that the information fed into the models was of the highest quality.

Model Development and Comparison

With a solid foundation of well-prepared data, we then focused on model development. We experimented with Random Forest, XGBoost, and LightGBM with optimized hyperparameters, and eventually landed on the best performing one.

Technical Specifications

Data Processing/Feature Engineering

These are the first steps we took in transforming our data:

For the turns level data we want to create a more comprehensive overview of game play, to do so we:

- One hot encode the 'turn_type' column
- Create a 'tiles_placed' feature that calculates the amount of letters added to the board for that turn from the user. This is easily calculated from our existing 'move' column, by simply removing the '.' from the move (the '.' represents existing letters on the board).
- Create a 'rare_letters' feature. We want to give an additional level of acknowledgment to moves that use tough letters. We define these tough letters as "Z", "Q", "J", "X", "K", "V", "Y", "W", and "G."
- Create a 'points_per_tile_placed' feature, which just takes the points feature and divides it by our newly defined 'tiles_placed' feature.
- Create 10 percentile wide categories for that turn's score. Using the overall distribution of points for all 2 million + turns, each turn is placed into one of these 10 percentile wide categories, therefore, each category is binary (0 or 1).
- Create a 'horizontal_placement' column. As we started to grasp a better understanding of the data set we came to realize that turns that started with a number were placed horizontally on the board. We add a binary variable to understand the direction the word is played.

One-Hot Encoding was necessary in this case since we must eventually aggregate all turn-level data into game-level data. Encoding scores into new turn_score percentile based groupings was entirely optional. However, we felt that this would, once aggregated, serve as a strong comparative measure for individual player scores at the game level.

Next we merged the train data that we were given that includes valuable information about what the bot and human were rated coming into the game with the game level data. This pregame rating is eventually going to be our target variable in our prediction. We want to split the bots and human players apart so that we are able to focus on creating human level features and bot level features and then merge them back together after doing so into one dataset.

At this point we were ready to aggregate our turn-level data.

Returning back to our turn level data, we want to aggregate this data set to have each row represent 1 single game. The logic behind this is so that we are able to combine it seamlessly with the game level data. To do so, we take a variety of statistics for each feature, by grouping by each game and within that game we take the following statistics for each feature:

Points:

- Mean
- Max
- Variance
- Min

Turn_Number:

- Count: This represents the total number of turns played by the user for that game

Turn_Type:

- Sum: Since Turn_Type is a binary variable, this represents the total number of that turn type played in that game
- Mean: Again, due to Turn_Type being binary, this represents the proportion of turns for that game that were this specific turn type.

Tiles_Placed:

- Mean
- Variance
- Max
- Min

Difficult_Letters:

- Sum: Represents the total number of difficult letters used in the game
- Mean: Represent the average amount of difficult letters played per turn in the game

Points_Placed_per_Tile:

- Mean
- Max
- Min
- Variance

Percentile Categories:

- Mean: Since these columns are all binary, this represents the proportion of turns in the game that were in this percentile category.
- Sum: This represents the total number of turns that fell into this percentile category in the game.

Horizontal_Placement:

- Sum: This represents the total number of horizontal words placed in the game.

Next, we split the bots and humans apart into two distinct datasets at the turn level. The reasoning behind this is that each game's turn is on a separate line, as the typical order of turns goes Bot, Human, Bot, Human…. We want to have the data condensed so that each game is 1 line. To do so we need to have the bots turn data and the players turn data in the same line next to each other

As mentioned, we now merge the human turn level data and the bots turn level data so that they are all in one row where each row represents an individual game.

Now that our turn-level data has been aggregated to be on the same level as the game level data, we can merge the aggregated turn level data with our original game level data, and One hot encode our categorical features from our original game level data:

- Time Control Name
- Game end reason
- Lexicon
- Rating Mode
- Bot name

Next we break up the data feature of the time the game was created at in to multiple time features:

- Year
- Month
- Day
- Hour
- Minute
- Second

We believe that by doing this, it will give us more detailed information about temporal habits of players.

Now that we have developed a much more detailed dataset to describe players performance for individual games, we want to look at game history. The reasoning behind this is that we are interested in predicting the players rating coming into the game, so it makes logical sense that if we look back in time and see how they were doing previously, our prediction should improve. To do this we will use SQL to develop 3 windows for calculating rolling averages for every column in our data set except for game_id and nickname. The 3 rolling averages we will look at are:

- Previous 3 games rolling average
- Previous 10 games rolling average
- Rolling average for all the games prior to that game for that individual in the data set.

Then, we simply take the results of that SQL query and make it our new data set for creating our models.

Next, we dropped the nickname column as this doesn't hold any predictive power for us. Again, as we mentioned previously, the rows with a null rating is the data that we will use as our training set. Therefore, we split the data into our testing set (rows with null values in rating) and our training set (rows without null values in rating).

We want to store game_id as a variable to eventually attach back to our predictions for submission to Kaggle, however, we want to remove it from the test_data because we do not want to feed it into our predictive modeling.

As a result of taking the rolling averages of the previous games played by the user, all the rows in our data set where it was the first game played by the user will now have null values for the rolling average features. Rather than dropping these rows and losing a lot of data, we decide to do KNN Imputation to fill the null values. We choose k = 5 to do so.

Now our data is ready for predictive modeling.

Predictive Models

Once our data was prepared, we decided to use three different models: XGBoost, LightGBM and Random Forest. XGBoost, LightGBM and Random Forest were all obvious choices for this project. This dataset is both large and highly complex. As a result, it stands to reason that a predictive model that is able to handle numerous types and facets of data well would be a good candidate. And due to the underlying structure of both XGBoost and LightGBM being based off of decision trees, we elected to try these models first. Since they all achieved such strong results, we did not feel the need to pursue other options.

Hyperparameter Tuning

Hyperparemeter tuning for these three models was a similar process for each, as they are all based off of a similar foundation of predictive process. That said, for our top model XGBoost we tuned the following hyperparameters:

- Max_depth: Sets the maximum depth of each tree. This helps the tree find balance between overfitting (with a large tree) and underfitting (with a small tree).
- Gamma: Determines what degree of loss is required to start a partition at a leaf node. Lower values of gamma lead to overfitting, while higher values can cause underfitting.
- Min_child_weight: Determines the minimum sum of instance weight needed in a given child. Larger values result in a more conservative algorithm.
- Subsample: The fraction of samples to be randomly sampled for each tree. Lower values result in a more conservative algorithm, but too small of values can cause underfitting.
- Colsample_bytree: The subsample ratio of columns for each tree. The lower this value is, the more conservative the model.
- Learning_rate: Controls step size shrinkage. Lower values slows the learning rate, reducing the risk of overfitting, but requiring more trees.
- Alpha: L1 regularization term (on weights). It encourages sparsity (zero values) which can serve as natural feature selection.
- Lambda: L2 regularization term (on weights). Tends to have a stronger penalty against complex models than L1.

During our tuning process, five-fold cross-validation was utilized to validate our models. An average RMSE was taken from each subset to estimate model performance. This process allowed us to mitigate overfitting of the model to the training data.

Hyperparameter tuning was performed through bayesian optimization. We chose this over a simple grid search due to its ability to evaluate a wide range of parameters efficiently. 5 iterations were used with 8 initial points.

Our optimal hyperparameters found for XGBoost were:

- Alpha: 0.6181
- Colsample_Bytree: 0.5802
- Gamma: 1.227
- Learning Rate: 0.09687
- Max_depth: 9.939

- Min_child_weight: 5.777
- N_estimators: 216
- Reg_lambda: 2.412

Similar methods of Bayesian optimization were also used for our other models. It is also worth noting that although XGBoost performed best out of the three models, that LightGBM performed far more efficiently, taking only 7.6 minutes to run compared to 46.6 minutes for XGBoost.

Results

	RMSE	Time-Consuming
XGBoost	98.757	46.6 minutes
LightGBM	99.58	7.6 minutes
Random Forest	103	1 hour

Summary & Forward Thinking

Summary

After multiple attempts, we've achieved our best result with a private score of 98.75 and a public score of 101.58. This marks a significant improvement from our initial score of 196. In this context, lower scores are better. From this project, we think there are two main factors that help our progress.

- Feature Engineering: Feature importance analysis provided by each model helps us understand which factors have the most significant impact on predicting performance ratings and identify key features that strongly influence a player's performance in Scrabble.

- Model Comparison: By comparing the performance of different models, such as XGBoost, LightGBM and Random Forest, we can get the best fit model for prediction.

Forward Thinking

This project also gives us some ideas how it can help businesses improve players' game experience.

- Player Segmentation: Identification of distinct player segments helps tailor marketing strategies, engagement initiatives and game features to specific players preferences and skill levels. It's like having a personalized approach for everyone.

- Dynamic Difficulty Adjustment: Using the models to adapt the game difficulty in real-time based on a player's predicted skill level ensures a challenging but enjoyable experience.

- In-Game Assistant: Providing real-time hints and challenges to a player's predicted skill level enhances their gaming experience. It's like having a virtual assistant within the game, providing just the right amount of guidance to enhance your overall gaming experience, which keeps the game both challenging and enjoyable.

- Educational and Training Modules: Scrabble, being a word game, involves language skills. The insights gained from successful gameplay can be used to develop educational modules within the game, offering language learning or cognitive training. Businesses can collaborate with educational institutions or language learning platforms to provide value-added content.
- Cross-Promotions with Other Products or Services: Analyzing player preferences and behaviors can uncover potential cross-promotion opportunities. Businesses can partner with other products or services that align with the gaming audience's interests, creating mutually beneficial marketing campaigns.
- Social Integration and Community Building: Scrabble victories can be shared on social media platforms. Integrating social features within the game, such as sharing achievements or challenging friends, can contribute to organic marketing and community building. Businesses can also capitalize on user-generated content for promotional purposes.

Appendix A: Full Engineered Feature List

Bot Features:

Features Built on Full History

10%-wide Categories of Turns

'avg_bot_0_10_percentile_0_10_percentile_mean_to_this_point',
'avg_bot_0_10_percentile_0_10_percentile_sum_to_this_point',
'avg_bot_10_20_percentile_10_20_percentile_mean_to_this_point',
'avg_bot_10_20_percentile_10_20_percentile_sum_to_this_point',
'avg_bot_20_30_percentile_20_30_percentile_mean_to_this_point',
'avg_bot_20_30_percentile_20_30_percentile_sum_to_this_point',
'avg_bot_30_40_percentile_30_40_percentile_mean_to_this_point',
'avg_bot_30_40_percentile_30_40_percentile_sum_to_this_point',
‘'avg_bot_40_50_percentile_40_50_percentile_mean_to_this_point',
'avg_bot_40_50_percentile_40_50_percentile_sum_to_this_point',
'avg_bot_50_60_percentile_50_60_percentile_mean_to_this_point',
'avg_bot_50_60_percentile_50_60_percentile_sum_to_this_point',
'avg_bot_60_70_percentile_60_70_percentile_mean_to_this_point',
'avg_bot_60_70_percentile_60_70_percentile_sum_to_this_point',
'avg_bot_70_80_percentile_70_80_percentile_mean_to_this_point',
'avg_bot_70_80_percentile_70_80_percentile_sum_to_this_point',
'avg_bot_80_90_percentile_80_90_percentile_mean_to_this_point',
'avg_bot_80_90_percentile_80_90_percentile_sum_to_this_point'
'avg_bot_90_100_percentile_90_100_percentile_mean_to_this_point'
'avg_bot_90_100_percentile_90_100_percentile_sum_to_this_point'
'bot_0_10_percentile_0_10_percentile_mean',
'bot_0_10_percentile_0_10_percentile_sum',
'bot_10_20_percentile_10_20_percentile_mean',
'bot_10_20_percentile_10_20_percentile_sum',
'bot_20_30_percentile_20_30_percentile_mean',
'bot_20_30_percentile_20_30_percentile_sum',
'bot_30_40_percentile_30_40_percentile_mean',
'bot_30_40_percentile_30_40_percentile_sum',
'bot_40_50_percentile_40_50_percentile_mean',
'bot_40_50_percentile_40_50_percentile_sum',
'bot_50_60_percentile_50_60_percentile_mean',
'bot_50_60_percentile_50_60_percentile_sum',
'bot_60_70_percentile_60_70_percentile_mean',
'bot_60_70_percentile_60_70_percentile_sum',
'bot_70_80_percentile_70_80_percentile_mean',
'bot_70_80_percentile_70_80_percentile_sum',
'bot_80_90_percentile_80_90_percentile_mean',
'bot_80_90_percentile_80_90_percentile_sum',
'bot_90_100_percentile_90_100_percentile_mean',
'bot_90_100_percentile_90_100_percentile_sum',

Difficult Letters, Horizontal Placement, Tiles Placed

'avg_bot_difficult_letters_difficult_letters_mean_to_this_point',
'avg_bot_difficult_letters_difficult_letters_sum_to_this_point',
'avg_bot_horizontal_placement_sum_to_this_point',
'avg_bot_tiles_placed_tiles_placed_max_to_this_point',
'avg_bot_tiles_placed_tiles_placed_mean_to_this_point',
'avg_bot_tiles_placed_tiles_placed_min_to_this_point',
'avg_bot_tiles_placed_tiles_placed_var_to_this_point',
'bot_difficult_letters_difficult_letters_mean',
'bot_difficult_letters_difficult_letters_sum',
'bot_horizontal_placement_sum',
'bot_tiles_placed_tiles_placed_max',
'bot_tiles_placed_tiles_placed_mean',
'bot_tiles_placed_tiles_placed_min',
'bot_tiles_placed_tiles_placed_var',

Points, Ratings, Scores, Turn Numbers, Bot Name

'avg_bot_points_per_tile_placed_points_per_tile_placed_max_to_this_point',
'avg_bot_points_per_tile_placed_points_per_tile_placed_mean_to_this_point',
'avg_bot_points_per_tile_placed_points_per_tile_placed_min_to_this_point',
'avg_bot_points_per_tile_placed_points_per_tile_placed_var_to_this_point',
'avg_bot_points_points_max_to_this_point',
'avg_bot_points_points_mean_to_this_point',
'avg_bot_points_points_min_to_this_point',
'avg_bot_points_points_var_to_this_point',
'avg_bot_rating_to_this_point',
'avg_bot_score_max_to_this_point',
'avg_bot_score_to_this_point',
'avg_bot_turn_number_count_to_this_point',
'avg_bot_name_BetterBot_to_this_point',
'avg_bot_name_HastyBot_to_this_point',
'avg_bot_name_STEEBot_to_this_point',
'bot_points_per_tile_placed_points_per_tile_placed_max',
'bot_points_per_tile_placed_points_per_tile_placed_mean',
'bot_points_per_tile_placed_points_per_tile_placed_min',
'bot_points_per_tile_placed_points_per_tile_placed_var',
'bot_points_points_max',
'bot_points_points_mean',
'bot_points_points_min',
'bot_points_points_var',
'bot_rating',
'bot_score',
'bot_score_max',
'bot_turn_number_count',
'bot_name_BetterBot',
'bot_name_HastyBot',
'bot_name_STEEBot',

Turn Types

'avg_bot_turn_type_Challenge_turn_type_Challenge_mean_to_this_point',
'avg_bot_turn_type_Challenge_turn_type_Challenge_sum_to_this_point',
'avg_bot_turn_type_End_turn_type_End_mean_to_this_point',
'avg_bot_turn_type_End_turn_type_End_sum_to_this_point',
'avg_bot_turn_type_Exchange_turn_type_Exchange_mean_to_this_point',
'avg_bot_turn_type_Exchange_turn_type_Exchange_sum_to_this_point',
'avg_bot_turn_type_Pass_turn_type_Pass_mean_to_this_point',
'avg_bot_turn_type_Pass_turn_type_Pass_sum_to_this_point',
'avg_bot_turn_type_Play_turn_type_Play_mean_to_this_point',
'avg_bot_turn_type_Play_turn_type_Play_sum_to_this_point',
'avg_bot_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_mean_to_this_point',
'avg_bot_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_sum_to_this_point',
'avg_bot_turn_type_Timeout_turn_type_Timeout_mean_to_this_point',
'avg_bot_turn_type_Timeout_turn_type_Timeout_sum_to_this_point',
'bot_turn_type_Challenge_turn_type_Challenge_mean',
'bot_turn_type_Challenge_turn_type_Challenge_sum',
'bot_turn_type_End_turn_type_End_mean',
'bot_turn_type_End_turn_type_End_sum',
'bot_turn_type_Exchange_turn_type_Exchange_mean',
'bot_turn_type_Exchange_turn_type_Exchange_sum',
'bot_turn_type_Pass_turn_type_Pass_mean',
'bot_turn_type_Pass_turn_type_Pass_sum',
'bot_turn_type_Play_turn_type_Play_mean',
'bot_turn_type_Play_turn_type_Play_sum',

'bot_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_mean',
'bot_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_sum',
'bot_turn_type_Timeout_turn_type_Timeout_mean',
'bot_turn_type_Timeout_turn_type_Timeout_sum',

Features Built on Last 10 Games

10%-wide Categories of Turns

'avg_last_10_bot_0_10_percentile_0_10_percentile_mean',
'avg_last_10_bot_0_10_percentile_0_10_percentile_sum',
'avg_last_10_bot_10_20_percentile_10_20_percentile_mean',
'avg_last_10_bot_10_20_percentile_10_20_percentile_sum',
'avg_last_10_bot_20_30_percentile_20_30_percentile_mean',
'avg_last_10_bot_20_30_percentile_20_30_percentile_sum',
'avg_last_10_bot_30_40_percentile_30_40_percentile_mean',
'avg_last_10_bot_30_40_percentile_30_40_percentile_sum',
'avg_last_10_bot_40_50_percentile_40_50_percentile_mean',
'avg_last_10_bot_40_50_percentile_40_50_percentile_sum',
'avg_last_10_bot_50_60_percentile_50_60_percentile_mean',
'avg_last_10_bot_50_60_percentile_50_60_percentile_sum',
'avg_last_10_bot_60_70_percentile_60_70_percentile_mean',
'avg_last_10_bot_60_70_percentile_60_70_percentile_sum',
'avg_last_10_bot_70_80_percentile_70_80_percentile_mean',
'avg_last_10_bot_70_80_percentile_70_80_percentile_sum',
'avg_last_10_bot_80_90_percentile_80_90_percentile_mean',
'avg_last_10_bot_80_90_percentile_80_90_percentile_sum',
'avg_last_10_bot_90_100_percentile_90_100_percentile_mean',
'avg_last_10_bot_90_100_percentile_90_100_percentile_sum',

Difficult Letters, Horizontal Placement, Tiles Placed

'avg_last_10_bot_difficult_letters_difficult_letters_mean',
'avg_last_10_bot_difficult_letters_difficult_letters_sum',
'avg_last_10_bot_horizontal_placement_sum',
'avg_last_10_bot_tiles_placed_tiles_placed_max',
'avg_last_10_bot_tiles_placed_tiles_placed_mean',
'avg_last_10_bot_tiles_placed_tiles_placed_min',
'avg_last_10_bot_tiles_placed_tiles_placed_var',

Points, Ratings, Scores, Turn Numbers, Bot Name

'avg_last_10_bot_points_per_tile_placed_points_per_tile_placed_max',
'avg_last_10_bot_points_per_tile_placed_points_per_tile_placed_mean',
'avg_last_10_bot_points_per_tile_placed_points_per_tile_placed_min',
'avg_last_10_bot_points_per_tile_placed_points_per_tile_placed_var',
'avg_last_10_bot_points_points_max',
'avg_last_10_bot_points_points_mean',
'avg_last_10_bot_points_points_min',
'avg_last_10_bot_points_points_var',
'avg_last_10_bot_rating',
'avg_last_10_bot_score',
'avg_last_10_bot_score_max',
'avg_last_10_bot_turn_number_count',
'avg_last_10_bot_name_BetterBot',
'avg_last_10_bot_name_HastyBot',
'avg_last_10_bot_name_STEEBot',

Turn Type

'avg_last_10_bot_turn_type_Challenge_turn_type_Challenge_mean',
'avg_last_10_bot_turn_type_Challenge_turn_type_Challenge_sum',
'avg_last_10_bot_turn_type_End_turn_type_End_mean',
'avg_last_10_bot_turn_type_End_turn_type_End_sum',
'avg_last_10_bot_turn_type_Exchange_turn_type_Exchange_mean',
'avg_last_10_bot_turn_type_Exchange_turn_type_Exchange_sum',
'avg_last_10_bot_turn_type_Pass_turn_type_Pass_mean',
'avg_last_10_bot_turn_type_Pass_turn_type_Pass_sum',
'avg_last_10_bot_turn_type_Play_turn_type_Play_mean',
'avg_last_10_bot_turn_type_Play_turn_type_Play_sum',
'avg_last_10_bot_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_mean',
'avg_last_10_bot_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_sum',
'avg_last_10_bot_turn_type_Timeout_turn_type_Timeout_mean',
'avg_last_10_bot_turn_type_Timeout_turn_type_Timeout_sum',

Features Built on Last 3 Games

10%-wide Categories of Turns

'avg_last_3_bot_0_10_percentile_0_10_percentile_mean',
'avg_last_3_bot_0_10_percentile_0_10_percentile_sum',
'avg_last_3_bot_10_20_percentile_10_20_percentile_mean',
'avg_last_3_bot_10_20_percentile_10_20_percentile_sum',
'avg_last_3_bot_20_30_percentile_20_30_percentile_mean',
'avg_last_3_bot_20_30_percentile_20_30_percentile_sum',
'avg_last_3_bot_30_40_percentile_30_40_percentile_mean',
'avg_last_3_bot_30_40_percentile_30_40_percentile_sum',
'avg_last_3_bot_40_50_percentile_40_50_percentile_mean',
'avg_last_3_bot_40_50_percentile_40_50_percentile_sum',
'avg_last_3_bot_50_60_percentile_50_60_percentile_mean',
'avg_last_3_bot_50_60_percentile_50_60_percentile_sum',
'avg_last_3_bot_60_70_percentile_60_70_percentile_mean',
'avg_last_3_bot_60_70_percentile_60_70_percentile_sum',
'avg_last_3_bot_70_80_percentile_70_80_percentile_mean',
'avg_last_3_bot_70_80_percentile_70_80_percentile_sum',
'avg_last_3_bot_80_90_percentile_80_90_percentile_mean',
'avg_last_3_bot_80_90_percentile_80_90_percentile_sum',
'avg_last_3_bot_90_100_percentile_90_100_percentile_mean',
'avg_last_3_bot_90_100_percentile_90_100_percentile_sum',

Difficult Letters, Horizontal Placement, Tiles Placed

'avg_last_3_bot_difficult_letters_difficult_letters_mean',
'avg_last_3_bot_difficult_letters_difficult_letters_sum',
'avg_last_3_bot_horizontal_placement_sum',
'avg_last_3_bot_tiles_placed_tiles_placed_max',
'avg_last_3_bot_tiles_placed_tiles_placed_mean',
'avg_last_3_bot_tiles_placed_tiles_placed_min',
'avg_last_3_bot_tiles_placed_tiles_placed_var',

Points, Ratings, Scores, Turn Numbers, Bot Name

'avg_last_3_bot_points_per_tile_placed_points_per_tile_placed_max',
'avg_last_3_bot_points_per_tile_placed_points_per_tile_placed_mean',
'avg_last_3_bot_points_per_tile_placed_points_per_tile_placed_min',
'avg_last_3_bot_points_per_tile_placed_points_per_tile_placed_var',
'avg_last_3_bot_points_points_max',
'avg_last_3_bot_points_points_mean',
'avg_last_3_bot_points_points_min',
'avg_last_3_bot_points_points_var',
'avg_last_3_bot_rating',
'avg_last_3_bot_score',
'avg_last_3_bot_score_max',
'avg_last_3_bot_turn_number_count',
'avg_last_3_bot_name_BetterBot',
'avg_last_3_bot_name_HastyBot',
'avg_last_3_bot_name_STEEBot',

Turn Type

'avg_last_3_bot_turn_type_Challenge_turn_type_Challenge_mean',
'avg_last_3_bot_turn_type_Challenge_turn_type_Challenge_sum',
'avg_last_3_bot_turn_type_End_turn_type_End_mean',
'avg_last_3_bot_turn_type_End_turn_type_End_sum',
'avg_last_3_bot_turn_type_Exchange_turn_type_Exchange_mean',

'avg_last_3_bot_turn_type_Exchange_turn_type_Exchange_sum',
'avg_last_3_bot_turn_type_Pass_turn_type_Pass_mean',
'avg_last_3_bot_turn_type_Pass_turn_type_Pass_sum',
'avg_last_3_bot_turn_type_Play_turn_type_Play_mean',
'avg_last_3_bot_turn_type_Play_turn_type_Play_sum',
'avg_last_3_bot_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_mean',
'avg_last_3_bot_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_sum',
'avg_last_3_bot_turn_type_Timeout_turn_type_Timeout_mean',
'avg_last_3_bot_turn_type_Timeout_turn_type_Timeout_sum',

Human Player Features:

Features Built on Full History

10%-wide Categories of Turns

'avg_0_10_percentile_0_10_percentile_mean_to_this_point',
'avg_0_10_percentile_0_10_percentile_sum_to_this_point',
'avg_10_20_percentile_10_20_percentile_mean_to_this_point',
'avg_10_20_percentile_10_20_percentile_sum_to_this_point',
'avg_20_30_percentile_20_30_percentile_mean_to_this_point',
'avg_20_30_percentile_20_30_percentile_sum_to_this_point',
'avg_30_40_percentile_30_40_percentile_mean_to_this_point',
'avg_30_40_percentile_30_40_percentile_sum_to_this_point',
'avg_40_50_percentile_40_50_percentile_mean_to_this_point',
'avg_40_50_percentile_40_50_percentile_sum_to_this_point',
'avg_50_60_percentile_50_60_percentile_mean_to_this_point',
'avg_50_60_percentile_50_60_percentile_sum_to_this_point',
'avg_60_70_percentile_60_70_percentile_mean_to_this_point',
'avg_60_70_percentile_60_70_percentile_sum_to_this_point',
'avg_70_80_percentile_70_80_percentile_mean_to_this_point',
'avg_70_80_percentile_70_80_percentile_sum_to_this_point',
'avg_80_90_percentile_80_90_percentile_mean_to_this_point',
'avg_80_90_percentile_80_90_percentile_sum_to_this_point',
'avg_90_100_percentile_90_100_percentile_mean_to_this_point',
'avg_90_100_percentile_90_100_percentile_sum_to_this_point',
'0_10_percentile_0_10_percentile_mean',
'0_10_percentile_0_10_percentile_sum',
'10_20_percentile_10_20_percentile_mean',
'10_20_percentile_10_20_percentile_sum',
'20_30_percentile_20_30_percentile_mean',
'20_30_percentile_20_30_percentile_sum',
'30_40_percentile_30_40_percentile_mean',
'30_40_percentile_30_40_percentile_sum',
'40_50_percentile_40_50_percentile_mean',
'40_50_percentile_40_50_percentile_sum',
'50_60_percentile_50_60_percentile_mean',
'50_60_percentile_50_60_percentile_sum',
'60_70_percentile_60_70_percentile_mean',
'60_70_percentile_60_70_percentile_sum',
'70_80_percentile_70_80_percentile_mean',
'70_80_percentile_70_80_percentile_sum',
'80_90_percentile_80_90_percentile_mean',
'80_90_percentile_80_90_percentile_sum',
'90_100_percentile_90_100_percentile_mean',
'90_100_percentile_90_100_percentile_sum',

Difficult Letters, Horizontal Placement, Tiles Placed

'avg_difficult_letters_difficult_letters_mean_to_this_point',
'avg_difficult_letters_difficult_letters_sum_to_this_point',
'avg_horizontal_placement_sum_to_this_point',
'avg_tiles_placed_tiles_placed_max_to_this_point',
'avg_tiles_placed_tiles_placed_mean_to_this_point',
'avg_tiles_placed_tiles_placed_min_to_this_point',
'avg_tiles_placed_tiles_placed_var_to_this_point',
'difficult_letters_difficult_letters_mean',
'difficult_letters_difficult_letters_sum',
'horizontal_placement_sum',
'tiles_placed_tiles_placed_max',
'tiles_placed_tiles_placed_mean',
'tiles_placed_tiles_placed_min',
'tiles_placed_tiles_placed_var',

Points, Ratings, Scores, Turn Numbers

'avg_points_per_tile_placed_points_per_tile_placed_max_to_this_point',
'avg_points_per_tile_placed_points_per_tile_placed_mean_to_this_point',
'avg_points_per_tile_placed_points_per_tile_placed_min_to_this_point',
'avg_points_per_tile_placed_points_per_tile_placed_var_to_this_point',
'avg_points_points_max_to_this_point',
'avg_points_points_mean_to_this_point',
'avg_points_points_min_to_this_point',
'avg_points_points_var_to_this_point',
'avg_score_max_to_this_point',
'avg_score_to_this_point',
'avg_turn_number_count_to_this_point',
'points_per_tile_placed_points_per_tile_placed_max',
'points_per_tile_placed_points_per_tile_placed_mean',
'points_per_tile_placed_points_per_tile_placed_min',
'points_per_tile_placed_points_per_tile_placed_var',
'points_points_max',
'points_points_mean',
'points_points_min',
'points_points_var',
'score',
'score_max',
'turn_number_count',

Turn Type

'avg_turn_type_Challenge_turn_type_Challenge_mean_to_this_point',
'avg_turn_type_Challenge_turn_type_Challenge_sum_to_this_point',
'avg_turn_type_End_turn_type_End_mean_to_this_point',
'avg_turn_type_End_turn_type_End_sum_to_this_point',
'avg_turn_type_Exchange_turn_type_Exchange_mean_to_this_point',
'avg_turn_type_Exchange_turn_type_Exchange_sum_to_this_point',
'avg_turn_type_Pass_turn_type_Pass_mean_to_this_point',
'avg_turn_type_Pass_turn_type_Pass_sum_to_this_point',
'avg_turn_type_Play_turn_type_Play_mean_to_this_point',
'avg_turn_type_Play_turn_type_Play_sum_to_this_point',
'avg_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_mean_to_this_point',
'avg_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_sum_to_this_point',
'avg_turn_type_Timeout_turn_type_Timeout_mean_to_this_point',
'avg_turn_type_Timeout_turn_type_Timeout_sum_to_this_point',
'turn_type_Challenge_turn_type_Challenge_mean',
'turn_type_Challenge_turn_type_Challenge_sum',
'turn_type_End_turn_type_End_mean',
'turn_type_End_turn_type_End_sum',
'turn_type_Exchange_turn_type_Exchange_mean',
'turn_type_Exchange_turn_type_Exchange_sum',
'turn_type_Pass_turn_type_Pass_mean',
'turn_type_Pass_turn_type_Pass_sum',
'turn_type_Play_turn_type_Play_mean',
'turn_type_Play_turn_type_Play_sum',
'turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_mean',
'turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_sum',
'turn_type_Timeout_turn_type_Timeout_mean',
'turn_type_Timeout_turn_type_Timeout_sum

Features Built on Last 10 Games

10%-wide Categories of Turns

'avg_last_10_0_10_percentile_0_10_percentile_mean',
'avg_last_10_0_10_percentile_0_10_percentile_sum',
'avg_last_10_10_20_percentile_10_20_percentile_mean',
'avg_last_10_10_20_percentile_10_20_percentile_sum',
'avg_last_10_20_30_percentile_20_30_percentile_mean',
'avg_last_10_20_30_percentile_20_30_percentile_sum',
'avg_last_10_30_40_percentile_30_40_percentile_mean',
'avg_last_10_30_40_percentile_30_40_percentile_sum',
'avg_last_10_40_50_percentile_40_50_percentile_mean',
'avg_last_10_40_50_percentile_40_50_percentile_sum',
'avg_last_10_50_60_percentile_50_60_percentile_mean',
'avg_last_10_50_60_percentile_50_60_percentile_sum',
'avg_last_10_60_70_percentile_60_70_percentile_mean',
'avg_last_10_60_70_percentile_60_70_percentile_sum',
'avg_last_10_70_80_percentile_70_80_percentile_mean',
'avg_last_10_70_80_percentile_70_80_percentile_sum',
'avg_last_10_80_90_percentile_80_90_percentile_mean',
'avg_last_10_80_90_percentile_80_90_percentile_sum',
'avg_last_10_90_100_percentile_90_100_percentile_mean',
'avg_last_10_90_100_percentile_90_100_percentile_sum',

Difficult Letters, Horizontal Placement, Tiles Placed

'avg_last_10_difficult_letters_difficult_letters_mean',
'avg_last_10_difficult_letters_difficult_letters_sum',
'avg_last_10_horizontal_placement_sum',
'avg_last_10_tiles_placed_tiles_placed_max',
'avg_last_10_tiles_placed_tiles_placed_mean',
'avg_last_10_tiles_placed_tiles_placed_min',
'avg_last_10_tiles_placed_tiles_placed_var',

Points, Ratings, Scores, Turn Numbers

'avg_last_10_points_per_tile_placed_points_per_tile_placed_max',
'avg_last_10_points_per_tile_placed_points_per_tile_placed_mean',
'avg_last_10_points_per_tile_placed_points_per_tile_placed_min',
'avg_last_10_points_per_tile_placed_points_per_tile_placed_var',
'avg_last_10_points_points_max',
'avg_last_10_points_points_mean',
'avg_last_10_points_points_min',
'avg_last_10_points_points_var',
'avg_last_10_score',
'avg_last_10_score_max',
'avg_last_10_turn_number_count',

Turn Type

'avg_last_10_turn_type_Challenge_turn_type_Challenge_mean',
'avg_last_10_turn_type_Challenge_turn_type_Challenge_sum',
'avg_last_10_turn_type_End_turn_type_End_mean',
'avg_last_10_turn_type_End_turn_type_End_sum',
'avg_last_10_turn_type_Exchange_turn_type_Exchange_mean',
'avg_last_10_turn_type_Exchange_turn_type_Exchange_sum',
'avg_last_10_turn_type_Pass_turn_type_Pass_mean',
'avg_last_10_turn_type_Pass_turn_type_Pass_sum',
'avg_last_10_turn_type_Play_turn_type_Play_mean',
'avg_last_10_turn_type_Play_turn_type_Play_sum',
'avg_last_10_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_mean',
'avg_last_10_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_sum',
'avg_last_10_turn_type_Timeout_turn_type_Timeout_mean',
'avg_last_10_turn_type_Timeout_turn_type_Timeout_sum',

Features Built on Last 3 Game

10%-wide Categories of Turns

'avg_last_3_0_10_percentile_0_10_percentile_mean',
'avg_last_3_0_10_percentile_0_10_percentile_sum',
'avg_last_3_10_20_percentile_10_20_percentile_mean',
'avg_last_3_10_20_percentile_10_20_percentile_sum',
'avg_last_3_20_30_percentile_20_30_percentile_mean',
'avg_last_3_20_30_percentile_20_30_percentile_sum',
'avg_last_3_30_40_percentile_30_40_percentile_mean',
'avg_last_3_30_40_percentile_30_40_percentile_sum',
'avg_last_3_40_50_percentile_40_50_percentile_mean',
'avg_last_3_40_50_percentile_40_50_percentile_sum',
'avg_last_3_50_60_percentile_50_60_percentile_mean',
'avg_last_3_50_60_percentile_50_60_percentile_sum',
'avg_last_3_60_70_percentile_60_70_percentile_mean',
'avg_last_3_60_70_percentile_60_70_percentile_sum',
'avg_last_3_70_80_percentile_70_80_percentile_mean',
'avg_last_3_70_80_percentile_70_80_percentile_sum',
'avg_last_3_80_90_percentile_80_90_percentile_mean',
'avg_last_3_80_90_percentile_80_90_percentile_sum',
'avg_last_3_90_100_percentile_90_100_percentile_mean',
'avg_last_3_90_100_percentile_90_100_percentile_sum',

Difficult Letters, Horizontal Placement, Tiles Placed

'avg_last_3_difficult_letters_difficult_letters_mean',
'avg_last_3_difficult_letters_difficult_letters_sum',
'avg_last_3_horizontal_placement_sum',
'avg_last_3_tiles_placed_tiles_placed_max',
'avg_last_3_tiles_placed_tiles_placed_mean',
'avg_last_3_tiles_placed_tiles_placed_min',
'avg_last_3_tiles_placed_tiles_placed_var',

Points, Ratings, Scores, Turn Numbers

'avg_last_3_points_per_tile_placed_points_per_tile_placed_max',
'avg_last_3_points_per_tile_placed_points_per_tile_placed_mean',
'avg_last_3_points_per_tile_placed_points_per_tile_placed_min',
'avg_last_3_points_per_tile_placed_points_per_tile_placed_var',
'avg_last_3_points_points_max',
'avg_last_3_points_points_mean',
'avg_last_3_points_points_min',
'avg_last_3_points_points_var',
'avg_last_3_score',
'avg_last_3_score_max',
'avg_last_3_turn_number_count',

Turn Type

'avg_last_3_turn_type_Challenge_turn_type_Challenge_mean',
'avg_last_3_turn_type_Challenge_turn_type_Challenge_sum',
'avg_last_3_turn_type_End_turn_type_End_mean',
'avg_last_3_turn_type_End_turn_type_End_sum',
'avg_last_3_turn_type_Exchange_turn_type_Exchange_mean',
'avg_last_3_turn_type_Exchange_turn_type_Exchange_sum',
'avg_last_3_turn_type_Pass_turn_type_Pass_mean',
'avg_last_3_turn_type_Pass_turn_type_Pass_sum',
'avg_last_3_turn_type_Play_turn_type_Play_mean',
'avg_last_3_turn_type_Play_turn_type_Play_sum',
'avg_last_3_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_mean',
'avg_last_3_turn_type_Six-Zero Rule_turn_type_Six-Zero Rule_sum',
'avg_last_3_turn_type_Timeout_turn_type_Timeout_mean',
'avg_last_3_turn_type_Timeout_turn_type_Timeout_sum',

Game Level Features:

One-hot-encoded Categorical Features

'avg_game_end_reason_CONSECUTIVE_ZEROES_to_this_point',
'avg_game_end_reason_RESIGNED_to_this_point',
'avg_game_end_reason_STANDARD_to_this_point',
'avg_game_end_reason_TIME_to_this_point',
'avg_rating_mode_CASUAL_to_this_point',
'avg_rating_mode_RATED_to_this_point',
'avg_lexicon_CSW21_to_this_point',
'avg_lexicon_ECWL_to_this_point',
'avg_lexicon_NSWL20_to_this_point',
'avg_lexicon_NWL20_to_this_point',
'avg_time_control_name_blitz_to_this_point',
'avg_time_control_name_rapid_to_this_point',
'avg_time_control_name_regular_to_this_point',
'avg_time_control_name_ultrablitz_to_this_point',

'game_end_reason_CONSECUTIVE_ZEROES',
'game_end_reason_RESIGNED',
'game_end_reason_STANDARD',
'game_end_reason_TIME',
'rating_mode_CASUAL',
'rating_mode_RATED',
'lexicon_CSW21',
'lexicon_ECWL',
'lexicon_NSWL20',
'lexicon_NWL20',
'time_control_name_blitz',
'time_control_name_rapid',
'time_control_name_regular',
'time_control_name_ultrablitz',

'avg_last_10_game_end_reason_CONSECUTIVE_ZEROES',
'avg_last_10_game_end_reason_RESIGNED',
'avg_last_10_game_end_reason_STANDARD',
'avg_last_10_game_end_reason_TIME',
'avg_last_10_rating_mode_CASUAL',
'avg_last_10_rating_mode_RATED',
'avg_last_10_lexicon_CSW21',
'avg_last_10_lexicon_ECWL',
'avg_last_10_lexicon_NSWL20',
'avg_last_10_lexicon_NWL20',
'avg_last_10_time_control_name_blitz',
'avg_last_10_time_control_name_rapid',
'avg_last_10_time_control_name_regular',
'avg_last_10_time_control_name_ultrablitz',

'avg_last_3_game_end_reason_CONSECUTIVE_ZEROES',
'avg_last_3_game_end_reason_RESIGNED',
'avg_last_3_game_end_reason_STANDARD',
'avg_last_3_game_end_reason_TIME',
'avg_last_3_rating_mode_CASUAL',
'avg_last_3_rating_mode_RATED',
'avg_last_3_lexicon_CSW21',
'avg_last_3_lexicon_ECWL',
'avg_last_3_lexicon_NSWL20',
'avg_last_3_lexicon_NWL20',
'avg_last_3_time_control_name_blitz',
'avg_last_3_time_control_name_rapid',
'avg_last_3_time_control_name_regular',
'avg_last_3_time_control_name_ultrablitz',

Time-relevant Features

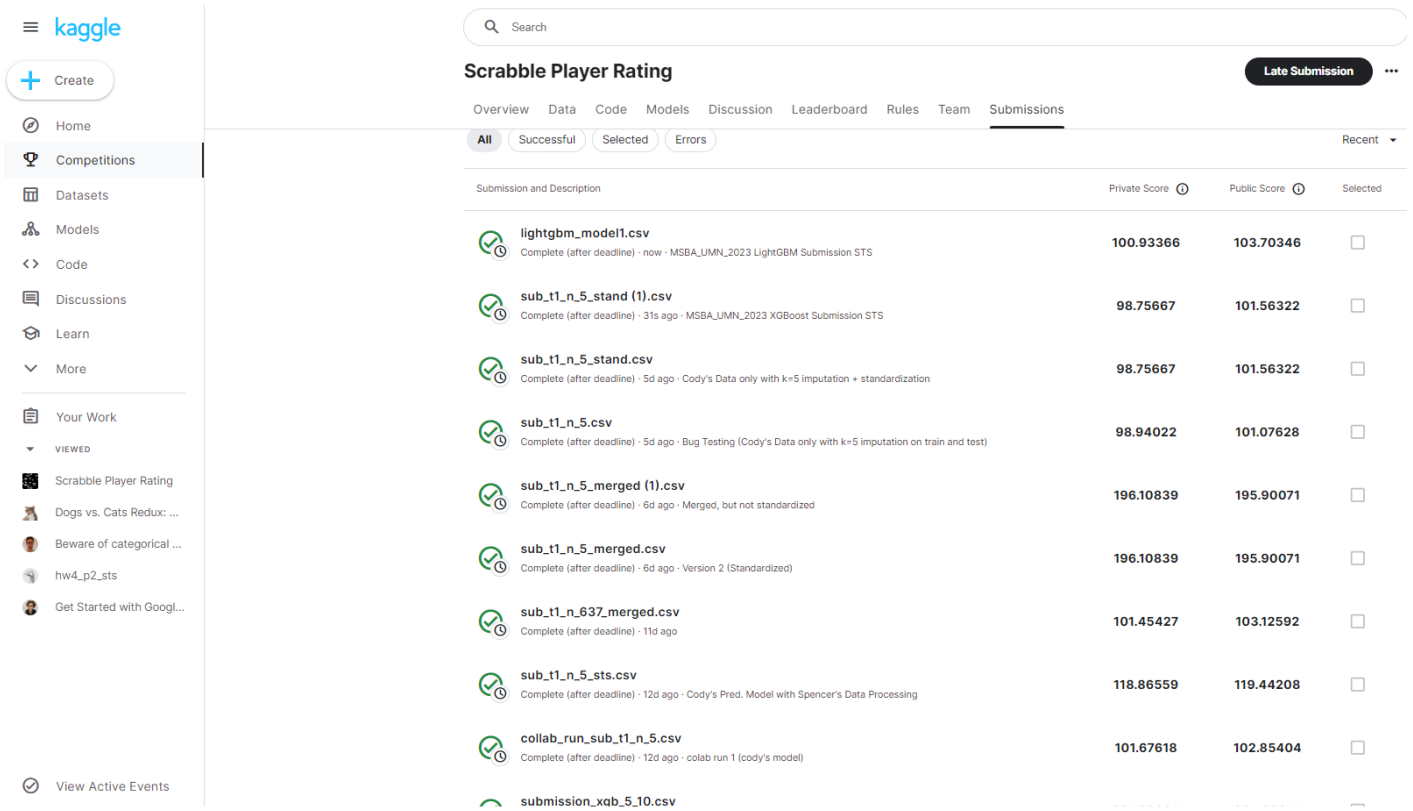
'avg_game_duration_seconds_to_this_point',
'avg_increment_seconds_to_this_point',
'avg_max_overtime_minutes_to_this_point',

'avg_initial_time_seconds_to_this_point',
'avg_seconds_to_this_point',
'avg_minute_to_this_point',
'avg_hour_to_this_point',
'avg_day_to_this_point',
'avg_month_to_this_point',
'avg_year_to_this_point',
'avg_first_to_this_point',
'avg_winner_to_this_point',

'game_duration_seconds',
'increment_seconds',
'initial_time_seconds',
'max_overtime_minutes',
'seconds',
'minute',
'hour',
'day',
'month',
'year'
'first',
'winner',

'avg_last_10_game_duration_seconds',
'avg_last_10_increment_seconds',
'avg_last_10_initial_time_seconds',
'avg_last_10_max_overtime_minutes',
'avg_last_10_seconds',
'avg_last_10_minute',
'avg_last_10_hour',
'avg_last_10_day',
'avg_last_10_month',
'avg_last_10_year',
'avg_last_10_first',
'avg_last_10_winner',

'avg_last_3_game_duration_seconds',
'avg_last_3_increment_seconds',
'avg_last_3_initial_time_seconds',
'avg_last_3_max_overtime_minutes',
'avg_last_3_seconds',
'avg_last_3_minute',
'avg_last_3_hour',
'avg_last_3_month',
'avg_last_3_year',
'avg_last_3_day',
'avg_last_3_first',
'avg_last_3_winner',


kaggle
Create
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work
VIEWED
Scrabble Player Rating
Dogs vs. Cats Redux: ...
Beware of categorical ...
hw4_p2_sts
Get Started with Googl...
View Active Events
Search
Scrabble Player Rating
Late Submission
Overview
Data
Code
Models
Discussion
Leaderboard
Rules
Team
Submissions
All
Successful
Selected
Errors
Recent
Submission and Description
Private Score
Public Score
Selected
lightgbm_model1.csv
Complete (after deadline) · now · MSBA_UMN_2023 LightGBM Submission STS
100.93366
103.70346
sub_t1_n_5_stand (1).csv
Complete (after deadline) · 31s ago · MSBA_UMN_2023 XGBoost Submission STS
98.75667
101.56322
sub_t1_n_5_stand.csv
Complete (after deadline) · 5d ago · Cody's Data only with k=5 imputation + standardization
98.75667
101.56322
sub_t1_n_5.csv
Complete (after deadline) · 5d ago · Bug Testing (Cody's Data only with k=5 imputation on train and test)
98.94022
101.07628
sub_t1_n_5_merged (1).csv
Complete (after deadline) · 6d ago · Merged, but not standardized
196.10839
195.90071
sub_t1_n_5_merged.csv
Complete (after deadline) · 6d ago · Version 2 (Standardized)
196.10839
195.90071
sub_t1_n_637_merged.csv
Complete (after deadline) · 11d ago
101.45427
103.12592
sub_t1_n_5_sts.csv
Complete (after deadline) · 12d ago · Cody's Pred. Model with Spencer's Data Processing
118.86559
119.44208
collab_run_sub_t1_n_5.csv
Complete (after deadline) · 12d ago · colab run 1 (cody's model)
101.67618
102.85404
submission_xgb_5_10.csv