SCRABBLE PROJECT

Yumi, Spencer, Cindy, Cody, Jessica

START

Agenda

Topic 1: Project Introduction

Topic 2: Solution Overview

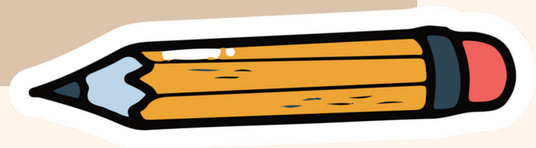
Topic 3: Data Processing & Exploration

Topic 4: Technical Specifications & Analysis

Topic 5: Conclusion & Forward Thinking

Project Introduction

This project aims to predict the ratings of human players on Woogles.io in Scrabble games using metadata and gameplay data. With ~73,000 games played by bots and humans, the task is to train a model on one set of opponents to predict ratings for a different set in the test data. The dataset includes game metadata, turns data, and pre-game ratings. *The objective is to forecast player ratings in the test set before the respective games were played.*



Solution Overview

1.

Data Processing

- Transform Turn and game level datasets
- Create Train and test dataset

2.

Visualization

- Explore the hidden patterns in the Turn data

3.

Feature Engineering

- Generate new aggregated features and rolling averages

4.

Predictive Model

- Feature importance ranking
- Test out different models and choose the best one

Data Overview

As part of this competition we were given game level data for 72,773 games, and for each of those games we had turn level data, for a total of 2,005,498 turns.

Game Data:

- game_id
- first
- time_control_name
- game_end_reason
- winner
- created_at
- lexicon
- initial_time_seconds
- increment_seconds
- rating_mode
- max_overtime_minutes
- game_duration_seconds

Turn Data:

- game_id
- turn_number
- nickname
- rack
- location
- move
- points
- score
- turn_type

Data Processing

Step 1:

Generating More Features from Turn Level Data

- Tiles Placed Feature:
  - Created 'tiles_placed' to show the number of letters used from a player's rack per turn.
- Difficult Letters Count:
  - Established 'difficult_letters' to count specific letter usage (Z, Q, J, X, K, V, Y, W, G).
- Word Placement Orientation:
  - Included a feature to identify word orientation as vertical or horizontal.
- One Hot Encoding for Turn Type:
  - Transformed 'turn_type' (6 categories: Challenge, End, Exchange, Pass, Play, Six-Zero Rule) into individual columns.
- Points per Tile Placed:
  - Calculated this feature by dividing turn points by the number of letters used from the rack.
- One Hot Encoding for Points Percentile:
  - Grouped turns into 10% point score percentiles for comparison against all dataset turns.

Data Processing

Step 2: Aggregating Turn Level Data

- Split Data:
  - Separate bots and human players into two distinct datasets.
- Group by 'game_id':

- Points Metrics:
  - Calculate mean, max, variance, and min of points.
- Turn Metrics:
  - Count the total number of turns per player.
  - Take the maximum score achieved.
- Turn Type Analysis:
  - For each turn type, compute:
    - Sum (total occurrences).
    - Mean (average occurrence rate).
- Tiles Placement Metrics:
  - Calculate mean, max, variance, and min for tiles placed.
- Difficult Letters Analysis:
  - Compute sum and mean for difficult letters.
- Percentile Category Analysis:
  - Take the sum and mean for each percentile category.
- Horizontal Placement:
  - Calculate sum and mean for horizontal placement.

Data Processing

Step 3: Game Level Data Conversion

- Combine user and bot level turn data into one row for each game.
- Merge this combined dataset with the game level data.

- Convert the following attributes into dummy variables:
  - time_control_name with 4 levels:
    - Blitz, Rapid, Regular, and Ultrablitz
  - game_end_reason with 4 levels:
    - Consecutive Zeros, Resigned, Standard, and Time
  - lexicon with 4 levels:
    - CSW21, ECWL, NSWL20, and NWL20
  - rating_mode with 2 levels:
    - Casual and Rated
  - bot_name with 3 levels:
    - BetterBot, HastyBot, and STEEBot
- Feature Creation:
  - Create individual features for each time unit:
    - Month, Day, Hour, Minute, Second

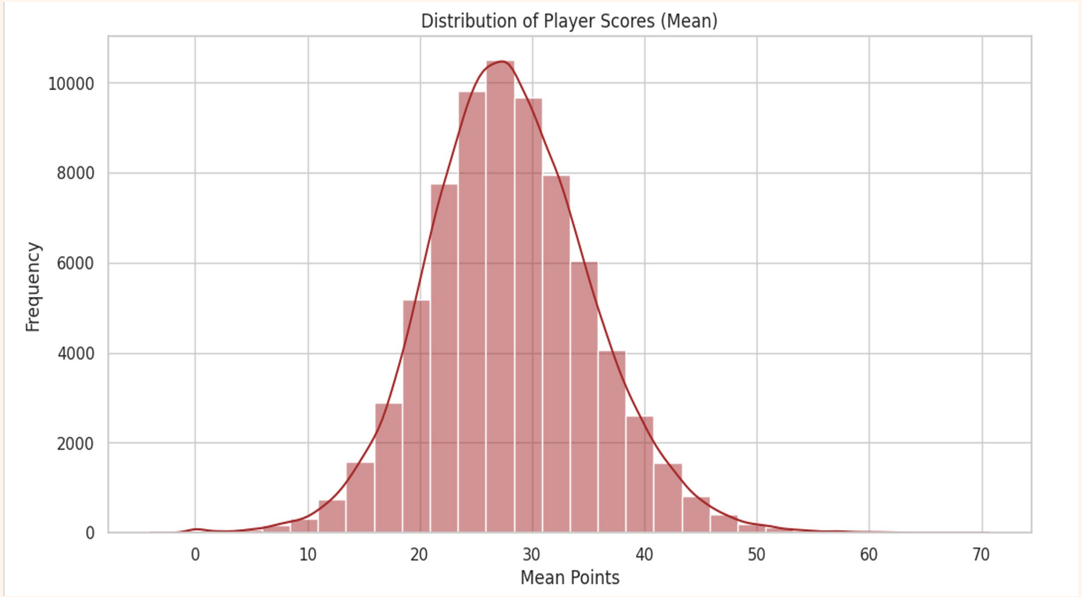
Exploration

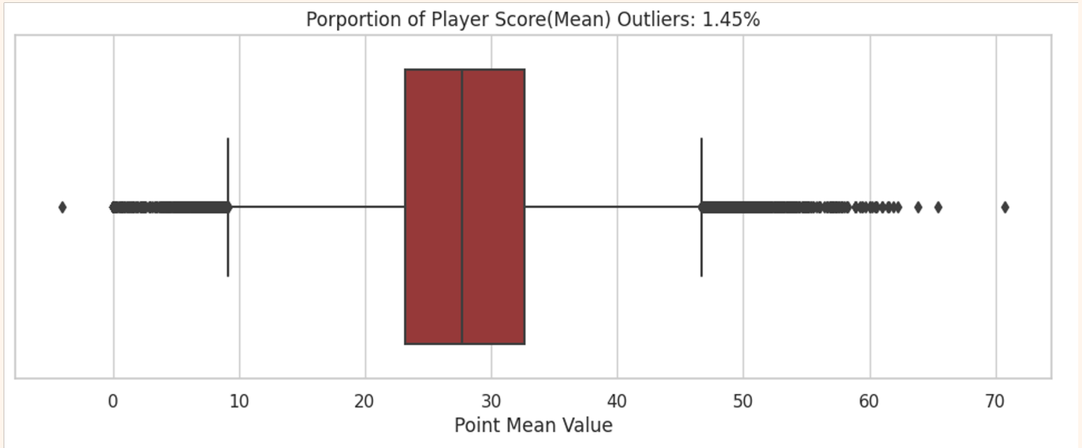
- Distribution of Player Scores(Mean)

  normal distribution
- Proportion of Player Score(Mean) Outliers: 1.93%

  Numbers of outliers: 1,944

  Numbers of non-outliers: 98,876

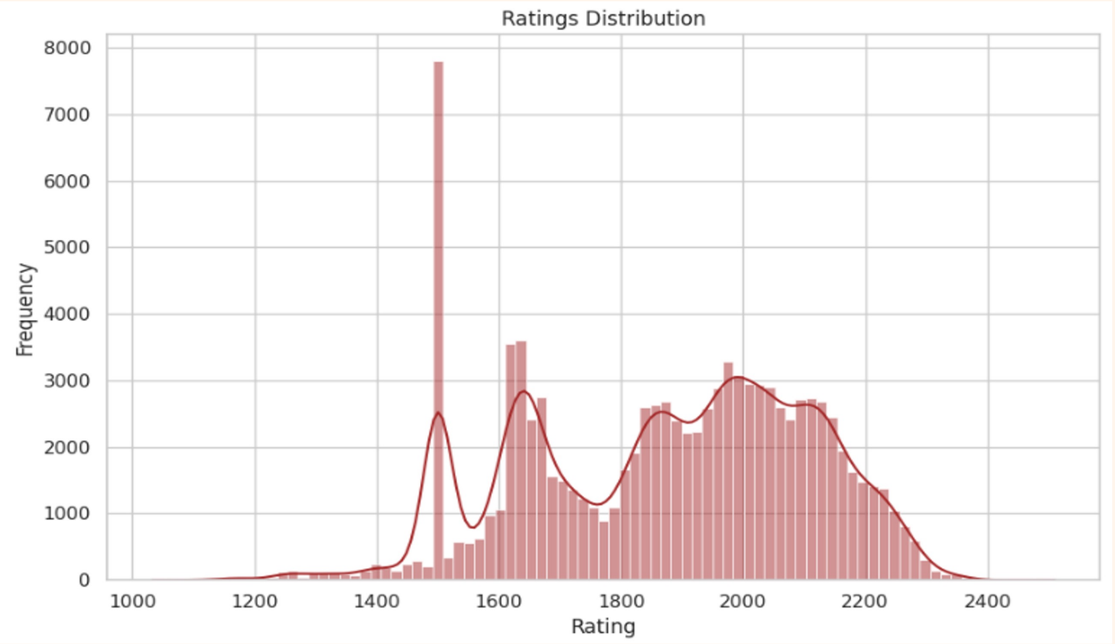


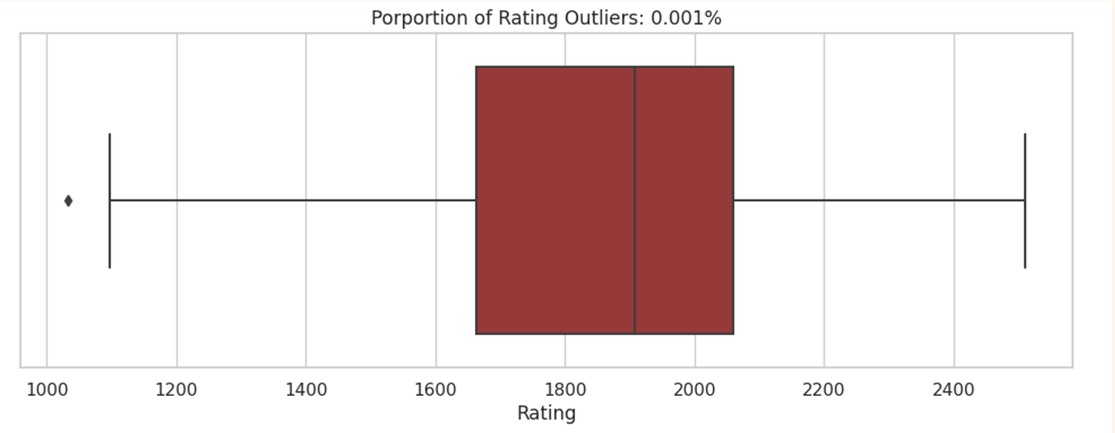


Exploration

- <u>Distribution of Ratings</u>
  - Most Ratings are 1,500
    - Default rating for first game

- <u>Proportion of Rating Outliers: 0.001%</u>
  - Highly Centralized





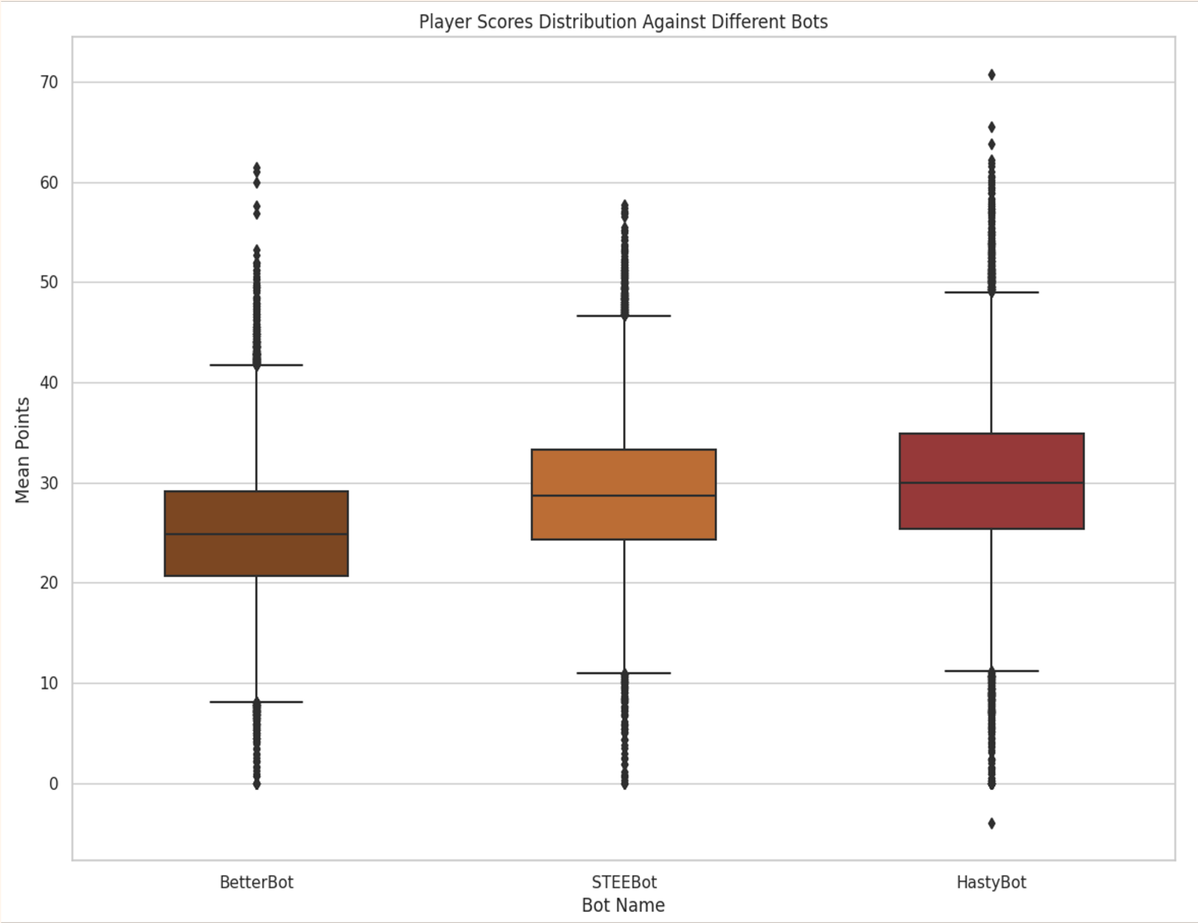
Exploration

- <u>Play Scores Distribution Against Different Bots</u>

  Highest: HastyBot

  Medium: STEEBot

  Lowest: BetterBot



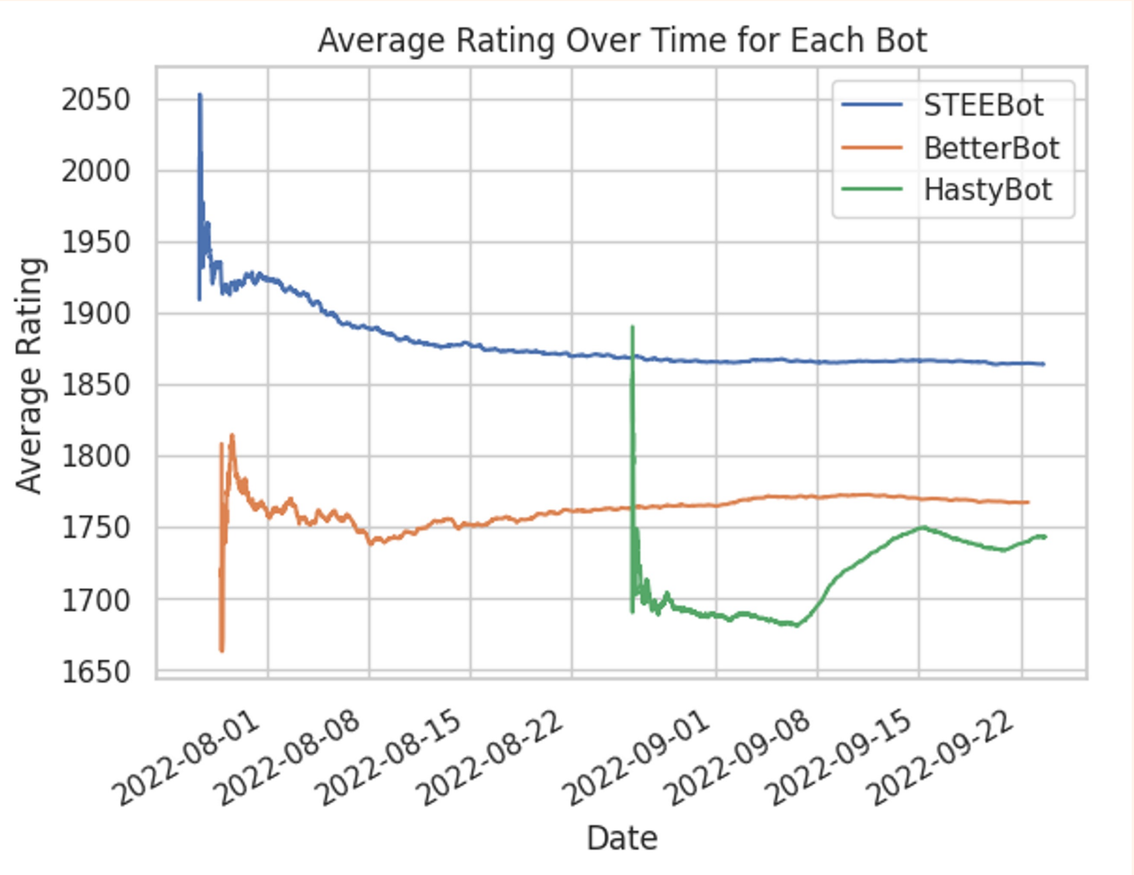
Exploration

- <u>Average Rating Over Time For Each Bot</u>

  Highest: STEEBot

  Medium: BetterBot

  Lowest: STEEBot



Feature Engineering

Feature Generation Strategy:

- Based on players' past game performances.
- Applied to every numerical column created during the data processing phase except the 'game_id' column.
- 542 total features

Rolling Average of the last 3 games played by the user

Rolling Average of the last 10 games played by the user

Rolling Average of all the previous games played by the user up until that point in time for the range given in the data set.

Then, since the first entry for each user did not have any previous games we used KNN Imputation to fill the NA values for the rolling averages

Feature Importance

1. Moving average of all previous games of the maximum score for a turn
2. Moving average over last 3 games of the proportion of games using lexicon_CSW21
3. Moving average of all previous games scores
4. Whether the current game is lexicon_CSW21
5. Moving average of the average points per turn of all previous games

Modeling

We used RMSE (Root Mean Squared Error) to compare the models performance:

- A larger RMSE means there's more variation in the prediction versus the reality
- A smaller RMSE means your prediction is closer to the reality

1

XGBoost

RMSE: 98.757

Time to Run: 46.6 minutes

2

LightGBM

RMSE: 99.58

Time to Run: 7.6 minutes

3

Random Forest

RMSE: 103

Time to Run: 1hr

Note: Time calculated from local computing environment with an Intel® Core™ i7 processor, 8GB of RAM, and a 64-bit operating system

Model Tuning Strategy - XGBoost

Hyperparameters

- max_depth
- gamma
- min_child_weight
- subsample
- colsample_bytree
- learning_rate
- alpha
- lambda

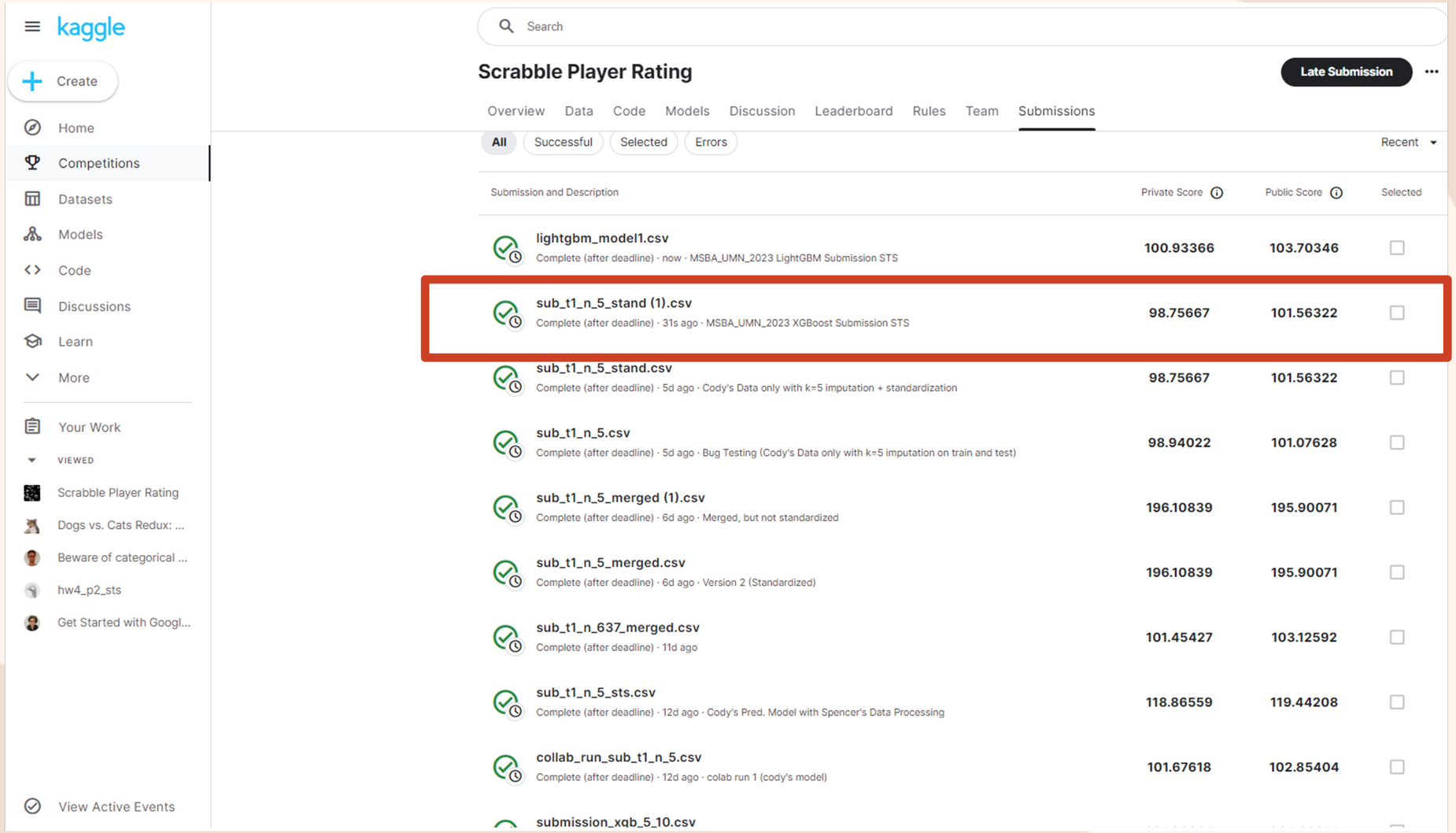
Avoid Overfitting

- Cross Validation
  - Validate model against 5 unique data subsets
  - Take average RMSE across each subset to estimate performance

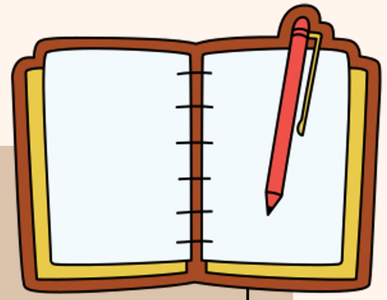
Hyperparameter Tuning

- Bayesian Optimization
  - Efficiently navigates wider range for hyperparameters

Result

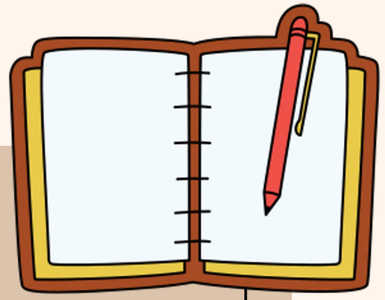

kaggle
Create
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work
VIEWED
Scrabble Player Rating
Dogs vs. Cats Redux: ...
Beware of categorical ...
hw4_p2_sts
Get Started with Googl...
View Active Events
Search
Scrabble Player Rating
Late Submission
Overview
Data
Code
Models
Discussion
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

Conclusion



- Feature Importance:
  Analyze feature importance provided by each model to 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:
  Compare the performance of XGBoost, LightGBM, and Random Forest models to get the best fit model for prediction.

Forward Thinking



- Player Segmentation:
  Identification of distinct player segments helps tailor marketing strategies, engagement initiatives and game features to specific players preferences and skill levels
- Dynamic Difficulty Adjustment:
  Use the models to adapt the game difficulty in real-time based on a player's predicted skill level, ensuring a challenging but enjoyable experience.
- In-Game Assistant:
  Providing real-time hints and challenges to a player's predicted skill level to enhance their gaming experience

THANK YOU SO MUCH!