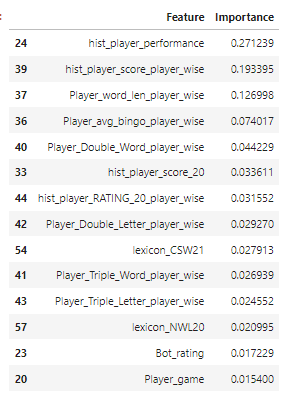
Team \_\_ : Praveen Kumar, Allison Liu, Harsha Podapati, Nick Seeman, Dev Rao

**Scrabble Report**

1. Project Definition
   1. Scrabble is a word game where players utilize letter tiles to create words on a board. The game can be played by 2,3, or 4 players, but in this project we are going to be looking at Scrabble being played by 2 players. Each player gets a set of 7 tiles which they can use each turn. The players take turns forming words from their set of tiles. Each letter tile has a point value, the goal of the game is to create words that maximize the total points. The game continues until all the tiles are used or players agree to end it. The winner is the player with the highest total score when the game concludes. Scrabble requires a combination of vocabulary and tile placement strategy. It is placed by all age groups for recreational and competitive purposes. In order to ensure consistent play amongst games, the Scrabble associations has a comprehensive word list that is common among all the games.
2. Project’s Goal
   1. In this project, based on Woogle.io datasets we are tasked with finding the rating of human players before a game is played. The dataset includes 72773 games played by three bots on Woogles.io BetterBot (beginner), STEEBot (intermediate), and HastyBot (advanced). The games are between the bots and their opponents who are regular users. We will be using metadata about the games and specific information about the turns in each game in order to predict the rating of the human opponents.
   2. Successful player rating will be measured by our predictions to real life player ratings provided in the test dataset. The rating provided from our model will be evaluated by RMSE
   3. The game board has a grid with special spaces that can multiply the point value of a letter or the entire word. Players draw letter tiles each turn to replenish their rack after each turn.
3. Technical Specification
   1. Tools
   2. Datasets
4. Data Prep
   1. EDA
   2. Outliers
5. Feature Engineering
6. Features Overview

Our dataset is game\_id base, so the following features all contain Bot and Player’s performance separately, ex: turns\_count included Bot\_turns\_count and Player\_turns\_count.

| Categoric | Features |  |
| --- | --- | --- |
| Game wise | turns\_count |  |
| max\_point |  |
| min\_point |  |
| avg\_bingo |  |
| avg\_word\_length |  |
| count\_letter\_JQXZ |  |
| rack\_used\_rate |  |
| win |  |
| lexicon\_CSW21 |  |
| lexicon\_NWL20 |  |
| Board wise | double\_word |  |
| triple\_word |  |
| double\_letter |  |
| triple\_letter |  |
| Player performance | score |  |
| rating |  |
| hist\_player\_score |  |
| hist\_player\_score\_20 |  |
| last\_game\_win\_flag |  |
| Player wise | avg\_bingo\_player\_wise |  |
|  | word\_len\_player\_wise |  |
|  | score\_player\_wise |  |
|  | hist\_player\_score\_player\_wise |  |
|  | double\_word\_player\_wise |  |
|  | triple\_word\_player\_wise |  |
|  | double\_letter\_player\_wise |  |
|  | triple\_letter\_player\_wise |  |
|  | hist\_player\_rating\_20\_player\_wise |  |

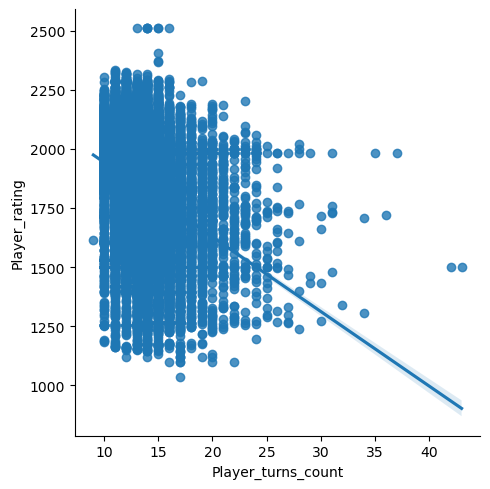


1. Game wise metrics
   * 1. Motivation

According to Turns.csv dataset, we found that there is a lot of information in between turns of each player in each game. Thus, we want to calculate and transform these columns into useful features.

* + 1. Turns\_count

We count how many times a player plays in each game. And, we found that the amount of players’ turns has moderate correlation to player rating, which means that players who have higher rating are using fewer turns.



* + 1. Max point / Min point

We calculate players’ maximum and minimum points in a game to state if he gets more points in each turn has an influence on rating score.

* + 1. Average bingo

Bingo is crucial for a player to get a high score and rating, so we calculate average bingos in each game.

* + 1. Average word length

The length of a letter is also important for players to arrange how to play and get high scores.

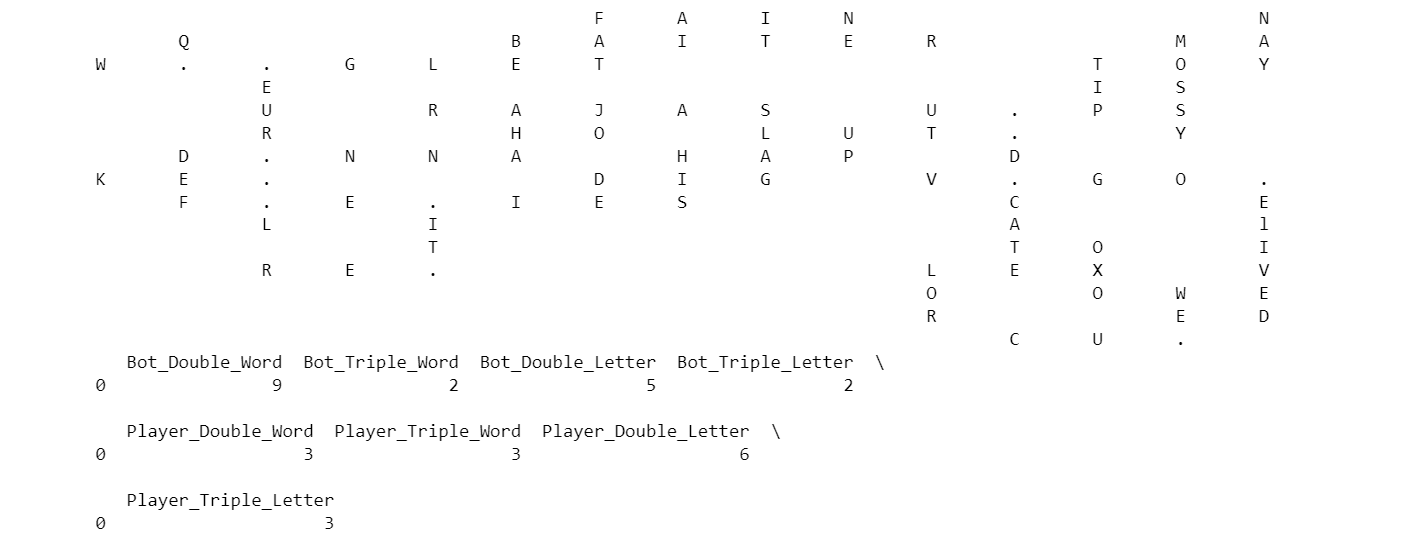
c. Board Constructions

* + 1. Motivation

We tried to work as an engine to simulate each turn for each player and games, and we found that how good a player is highly related to the strategies they touch the special board squares, ex: double-word squares and double-letter squares.

* + 1. Board metrics

We simulated the board game using ‘location’ information in the turns.csv dataset to build a board.



* + 1. Special Moves

We know that if words or letters were placed on the specific tiles - double-word, triple-word, double-letter, triple-letter squares. Thus, we calculate how many special moves that players achieved to evaluate players’ level.

* 1. Player performance metrics
     1. Motivation

We knew that most of the games in the sets were from **frequently occurring players**, so featuring the player’s history was crucial.

* + 1. Historic player performance / previous 10 or 20 games

**Rating: Average Opponent Rating + 400 \* (Wins-Losses) / Games**

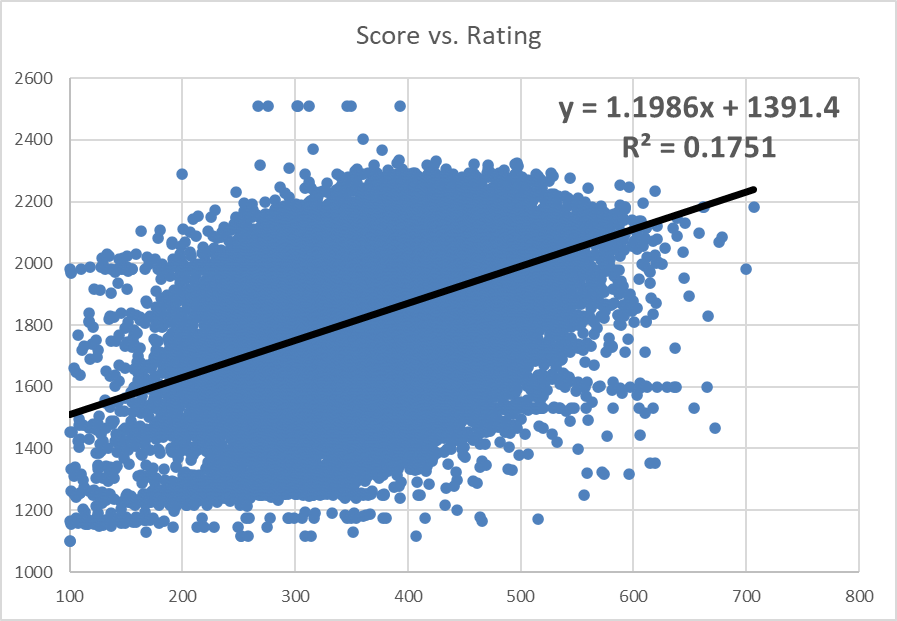
We calculated players’ historic performances from their previous 10 and 20 games to understand that their recent performances influenced current rating. The basis for this formula determines player ratings on the site, so it is a direct measure of the rating we are trying to estimate!

* 1. Player wise metrics
     1. Motivation

While every player has a unique play style, good players tend to do good things in the game more often than bad players. We sought to find indications of good play within the game, as a method to diversify features.

* + 1. Historical Performances ???? do we mean something else?????

Players who score more points more often tend to win more often. A very direct attempt to solve our problem here begins with a simple regression of player rating vs. player score. The data has quite a bit of noise, but the overall trend is obvious. Roughly, each additional point a player scores on average correlates to 1.2 extra rating points.



One key to this is that the strength of the opponent has an impact on the player’s ability to score points as well. Stronger opponents are able to capitalize on high-scoring squares for themselves, and block those squares for their counterparts. Directly measuring the blocking is difficult, as it requires deep in-game strategic analysis, so we focused merely on the aspects within the player’s control during their turn.

The in-game metrics that we calculated, such as word length, bingos, etc., were averaged for all games. This helps reduce the variance of these metrics for a player, as some of these events do not happen frequently within each game. Getting a broader sense of the players’ skill was most important for this task, therefore by using more information (outside of the individual game we are estimating) about the player we found better model performance.

1. Algorithm Selection & Parameter Tuning

We explored multiple predictive models starting from basic or conventional algorithms like Multiple linear regression, Logistic regression, decision tree regressor and moved on fairly advanced algorithms like Random Forest Regressor and to ensemble models like XGBoost, Light GBM, further extended to deep neural network as well. Here are the top observations.

* 1. Model Explorations

In the algorithm selection process, we tried XGBoostRegressor, LightGBMRegressor, Random Forest, Neural network, and ensemble models of random forest and XGBoost. We will further conduct parameter

tuning for these models.

* + 1. Random Forest

The best parameters for Random forest:

'max\_depth': [16],

'n\_estimators': [ 300],

'max\_features': ['sqrt'],

'bootstrap': [False]

Cv = 3

Much of the model selection process was clouded by the big differences between validation scores and test scores. The models were overfitting the data due to the nature of the data. However, testing data consistently was better for higher depth forests. Larger forests did not perform any better in the training and validation portion. We found limiting the max features resulted in slightly better performance.

K-Fold Cross-Validation evaluation:

* + 1. Xgboost

The best parameters for Xgboost:

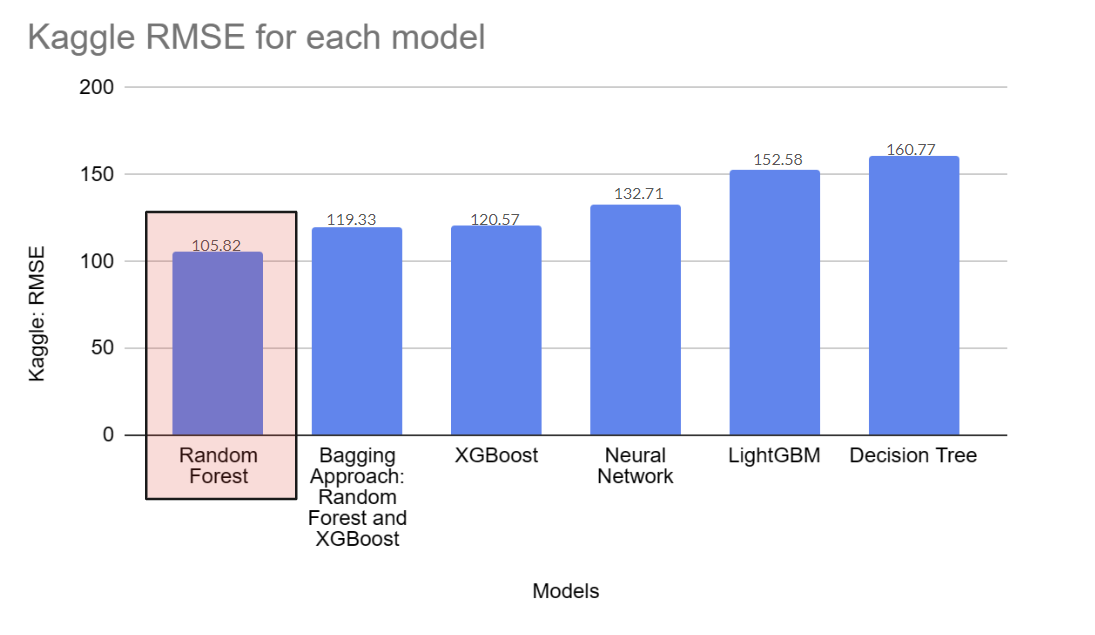


K-Fold Cross validation evaluation:



* + 1. Lightgbm
    2. Decision Tree
    3. Neural Networks
  1. Model Selection

Cross-validation: Grid-Search



* 1. Best model explanation

Hyperparameters:

Max-Depth: 8

Minimum Sample Leaf: 1

Minimum Sample Split: 5

Number of Estimators: 50

1. Evaluation
2. Future Improvement

**Big Data**

A future improvement we would like to implement to our model would be to take it onto a big data software. We would like our model to be able to generate predictions as new data comes into it as that is how game websites do it. Doing this would make our model equivalent to a normal predictive game ratings model used in games like Scrabble on Woogles.io.

**Increased Model Explorations**

A future improvement we would like to explore is whether models different from the ones we made would work better. We found that the Random Forest model was the best of the group we chose to compare, but we believe there may be a way to improve the current Random Forest model or find a different model that works better. The model of interest to us is the Neural Network, even though we have evaluated one Neural Network model already. We are interested in the Neural Network because of its typically high performance. It is interesting to explore further to see if we get a neural network model that outperforms the random forest model we made.

**Add More Features**

A future improvement we would like to explore is creating even more additional features. During our project, we created 28 features to help build our model, but it is possible to develop more features. Creating more features based on historical data could increase our model’s ability to predict player ratings even more than the current model.

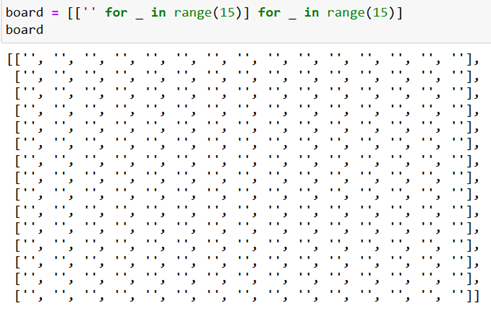
**Build the scrabble board:** From the “turns” dataset, we have details of every move played between the player and the bot turn wise for a particular game. We built the simulation of this board play for every game to extract board level metrics. Let’s look at how to construct the board from the turns dataset.

Turns dataset has the column called move which got the information of the set of letters they use for the play, it is the move they play, for example: could be the word “DIG”, next, turns dataset also has this column called location which gives the information about the position at which the move is played.

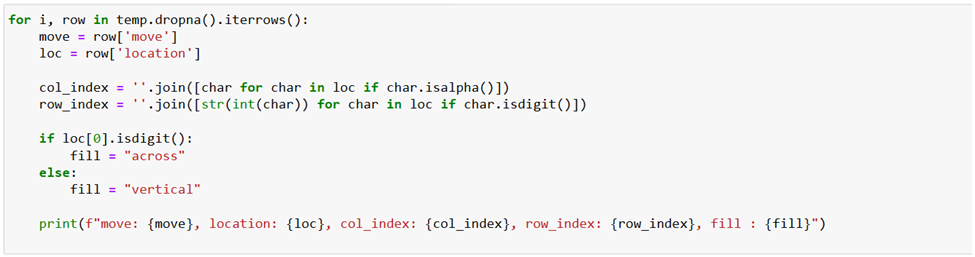
Interestingly, in scrabble, it is a 15 x 15 board, where we have the rows as numbers and columns as alphabets, the way or the location values are filled denotes whether it is filled across or vertically. Below are the steps that were carried out to build the board.

*Step 1: Initialize the board:*

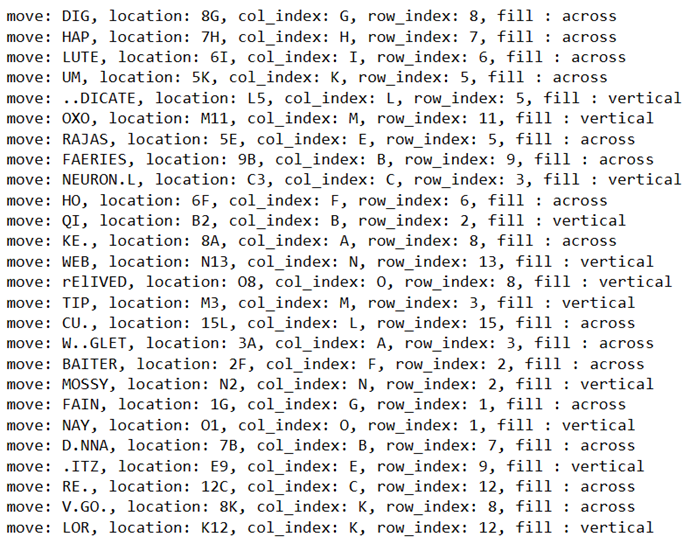
This line creates a 2D list (board) initialized with empty strings. It represents a 15x15 crossword puzzle board.



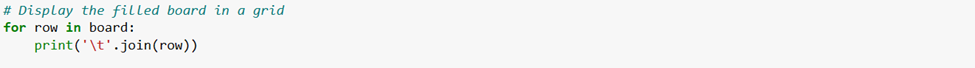
*Step 2: This loop iterates over non null rows and extracts information about the move (word) and its location from the current row.*

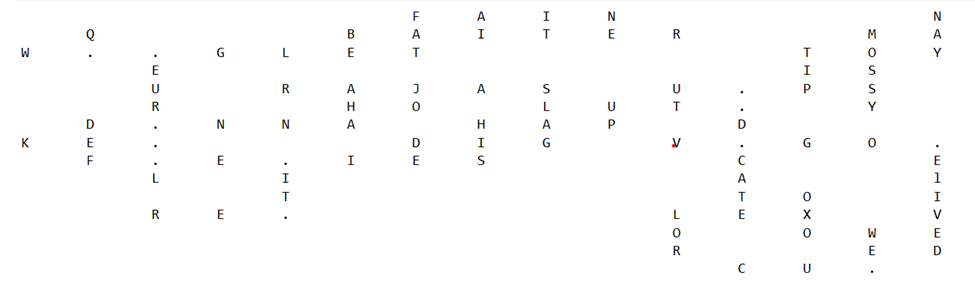
**

Col\_index, row\_index extracts the column index (letters) and row index (numbers) from the location and the way that location variable is arranged determines how the word is filled whether across (horizontal) or vertical.

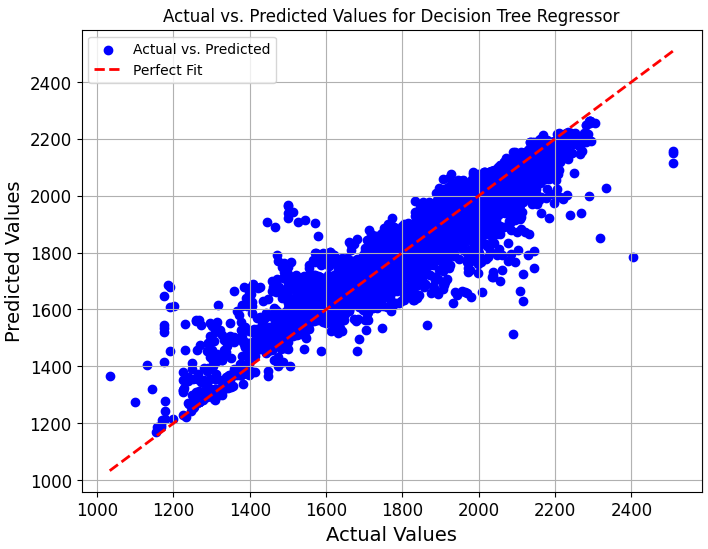


*Step 3: Display and save the board as a 2D matrix.*

**

**

Why was the board built for every game played, what value are we trying to generate out of it, will be discussed in the subsequent section of the document.



Training data, model, and validation. Test data is not available for this graph.

**Business Values**

Several real-world business applications exist for the model used here, or at least the techniques used to solve this problem. Our most direct example is the obvious crossover into sports, and in particular, sports betting and gambling. With analytics being at the forefront of most sports, we are able to collect lots of data and information not only about the games themselves, but also about individual player statistics.

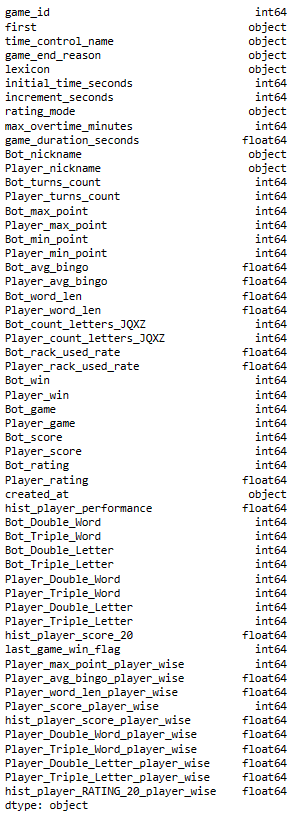
A modification to the model to determine probability of a win could allow a bettor to directly bet on the moneyline of games. A few similar modifications could be used to determine the appropriate spread of a game, or even the total points scored in a game. These 3 bets are by far the most common ways to bet on sports events for sports like basketball. While an individual can choose to do this for themselves, there are also many large sportsbooks who are looking to set good lines and maintain their house advantage. Risk management for these casinos employ machine learning models to set competitive lines and improve profitability.

Franchises and teams have long employed data scientists to help make roster improvements and build the best teams they can – or at least at the prices they can afford. Our task for this model only involved data for two months. However, if enough data was collected over a longer time horizon, we believe it to be possible for a model like this to help predict the future abilities of a player in a game. This is likely the most valuable aspect for a franchise, as signing young and talented players is often the cheapest way to be successful.

Customer habits and behavior prediction could also be modeled with the techniques we have used here. Several aspects of consumer loans garner lots of attention and labor from banks: credit/default risk, and prepayment risk. These are both behaviors which can be, and are, modeled extensively at banks and investment shops of all sizes.

**Feature Selection:**

After immense efforts to capture the nuances of the scrabble game at levels of board usage, goodness of play through turns and lastly historic player level performance, we arrive at these sets of features as a whole. To avoid the curse of dimensionality, we performed feature selection on top of all the engineered features and got the data ready for predictive modeling.



One hot encoding the category variables:



Random Forest for feature selection:

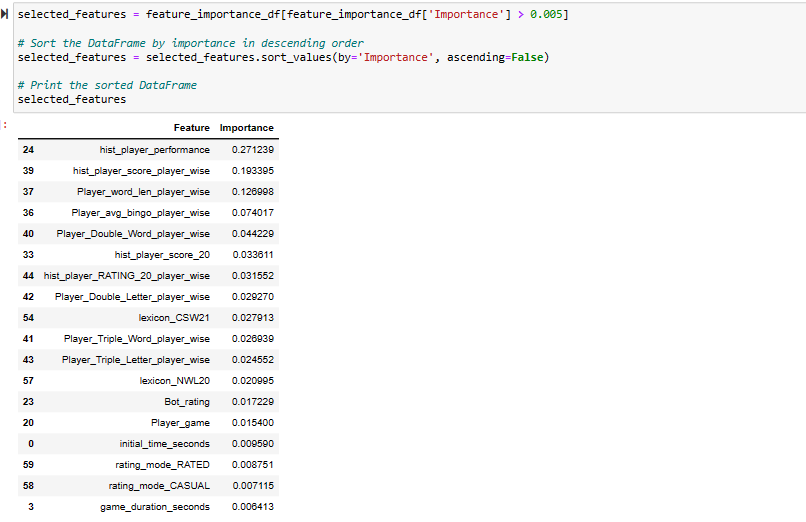
The following code utilizes a RandomForestRegressor from the scikit-learn library to predict the 'Player\_rating' target variable based on a set of features. The features include various aspects related to the gameplay, such as time control settings, turn counts, word lengths, rack utilization rates, and other relevant parameters.

The RandomForestRegressor is trained on the entire dataset (df), where 'X' represents the feature matrix (all columns except 'Player\_rating'), and 'y' represents the target variable ('Player\_rating'). The model is then fitted to the data, and the feature importances are calculated, reflecting the contribution of each feature to the prediction.

The code creates a DataFrame (feature\_importance\_df) to store the computed feature importances, with columns for feature names and their corresponding importance scores. The DataFrame is sorted in descending order based on feature importance, and the results are printed.

Furthermore, the code identifies features with importances greater than 0.005, resulting in a subset of selected features (selected\_features). This subset is sorted by importance, and the resulting DataFrame is printed. The selected feature names are extracted, and a new DataFrame (df\_selected) is created by selecting columns corresponding to the chosen features and adding the 'Player\_rating' column.





Decision Trees:

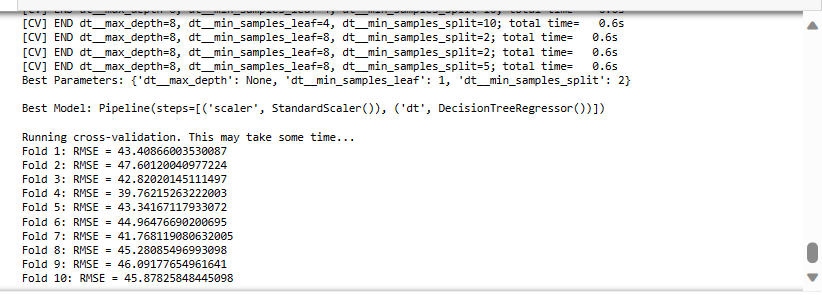
In this following code, a machine learning pipeline was constructed for predicting player ratings using a Decision Tree Regressor. The dataset, denoted as df\_selected, is feature selected at earlier stages with the selected features stored in X and the corresponding player ratings in y. The pipeline incorporated a StandardScaler for feature scaling.

To optimize the model's performance, a grid search was conducted using GridSearchCV. The grid search involved exploring different hyperparameter combinations for the Decision Tree Regressor, specifically varying max\_depth, min\_samples\_split, and min\_samples\_leaf. The performance metric used for evaluation was the Root Mean Squared Error (RMSE), configured through the creation of a custom scoring function.

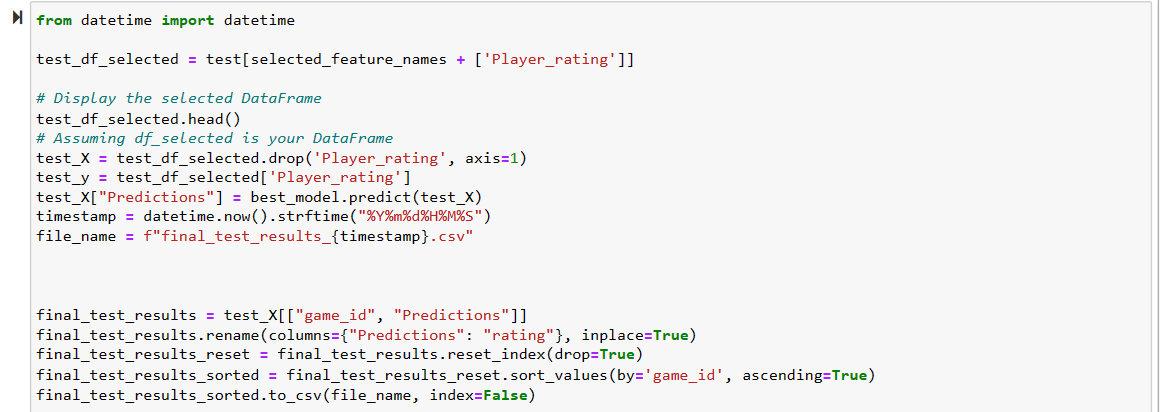
The results of the grid search indicated the optimal hyperparameters for the Decision Tree Regressor. The best configuration was found to be with no specified maximum depth (None), a minimum number of samples per leaf set to 1, and a minimum number of samples required to split an internal node set to 2.

Subsequently, the best model, identified by the grid search, was printed along with its associated parameters. A 10-fold cross-validation was performed to assess the model's generalization performance. The RMSE was calculated for each fold, revealing the model's predictive accuracy across different subsets of the dataset.





Predictions were generated to check the model performance in kaggle out of sample test data:



Performance:

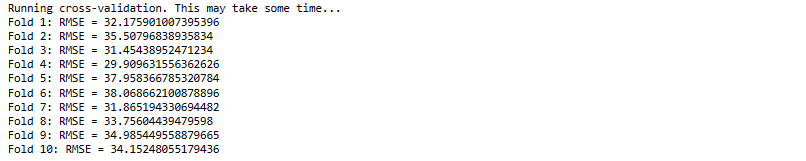


Light GBM:

In the following code, a LightGBM regression model is trained on a dataset (df\_selected) to predict the 'Player\_rating' target variable. The model is built using a pipeline that includes a StandardScaler for feature scaling and the LightGBMRegressor as the underlying regressor. A grid search is conducted to find the best hyperparameters for the model, considering variations in the number of estimators, maximum depth, learning rate, minimum child samples, subsample, and colsample\_bytree.

The grid search is performed using 10-fold cross-validation, and the root mean squared error (RMSE) is chosen as the evaluation metric. The cross-validation process involves fitting the model on different training sets and evaluating its performance on corresponding test sets. The printed output displays the RMSE for each fold, providing insights into the model's generalization performance across different subsets of the data.







SCRABBLE VIDEO: <https://www.youtube.com/watch?v=zw5c425LxMs>