"Exploring Online Chess Games Using a Data Science Based Approach"

Yadvender Singh
50466664
yadvende@buffalo.edu
Computer Science and Engineering
CSE 587
University at Buffalo
Buffalo, New York, USA

Shawna Saha
50468278
shawnasa@buffalo.edu
Computer Science and Engineering
CSE 587
University at Buffalo
Buffalo, New York, USA

I. Introduction

Since time immemorial Chess has been a prevalent game that has captivated the minds of a large audience. The highly esoteric nature of this game, along with its simple rules but overly complex and intricate plays make it an ideal sport for information extraction via data analysis. Data Science can be applied to chess game analysis to build player profiles, build fraud detection system in online chess games, recommender system for chess training and practice etc.

II. Problem Statement

Though the game of chess has been played for millennia, there is a lack of comprehensive understanding of the dynamics of the game. Determining the optimum set of moves to achieve victory is a significant struggle since there is no one right strategy to play the game. The game's arcane nature has made it a challenge even for the brightest brains.

With the development of technology acquiring data on chess games have become relatively feasible. By analyzing the enormous databases of chess games, data science approaches can help in spotting patterns and trends, create models that may forecast upcoming moves and results.

Thus, this project aims to employ data science techniques in the analysis of a large dataset of online chess games to gain insights into the patterns, trends and strategies used by different chess players at different levels of the game.

The primary questions we tried to answer through exploratory data analysis were:

1. What are the most common openings used by successful chess players and how does it affect their game?
2. Is it possible to predict the outcome of a game based on the opening move chosen by a player?
3. What sequence of moves have the largest probability to win?
4. How does a player's rating affect the choice of opening move and the outcome of the game?
5. How do players of similar ratings perform when played against each other?
6. Does a white player have an advantage in how the game unfolds due to the rule of the game that states – "White moves first"?

Now, by employing several machine learning algorithms, we are trying to make predictions about which player will emerge victorious in a game of chess.

Answering these questions will help in attaining the following objectives:

1. By analyzing the data to better understand a player's performance such as – number of wins, number of losses, preferred first move etc. can help in coaching and suggest improvements to the game.
2. Insights into the most common opening strategies, number of turns, sequence of moves, the most successful tactics etc. can help in creating better training materials.

III. Data Sources

The dataset used for the implementation of the project are detailed below:
Name of the dataset: Chess Game Dataset (Lichess)
Source: www.kaggle.com
Link: https://www.kaggle.com/datasets/datasnaek/chess?datasetId=2321&searchQuery=eda

Description: This is a dataset containing data collected from 20,000 online chess games from the site Litchess.org. The dataset has 16 features in total. They are as detailed below:

1. ID: Game id.
2. RATED: If a player is rated or not.
3. CREATED AT: Start time of the game.
4. LAST MOVE AT: End time of the game.
5. TURNS: Number of turns in the game.
6. VICTORY STATUS: The outcome of the game.
7. WINNER: Which player won.
8. INCREMENT CODE: The amount of time added to the clock after each move made by a player.
9. WHITE ID: Id of the white player.
10. WHITE RATING: Rating of the white player.
11. BLACK ID: Id of the black player.
12. BLACK RATING: Rating of the white player.
13. MOVES: Set of moves made by the players in chess notation.
14. OPENING ECO: Standardized codes for the chess opening names.
15. OPENING NAME: The chess opening used by the player.
16. OPENING PLY: Number of moves in the opening phase.

IV. ALGORITHMS

The machine learning algorithms we will be using to make predictions about the winner of a chess game are as follows:

Algorithms discussed in class:

1. Naïve Bayes
2. K-Nearest Neighbor
3. Logistic Regression

Algorithms outside of class discussion:

4. Decision Tree
5. Random Forest
6. Neural Network - Multi-Layer Perceptron

V. DATA PRE-PROCESSING FOR MODEL FITTING

The following pr-processing steps were followed to process the data further in order to fit the machine learning models.

a. **Dropping "draw" outcomes:** Since we are only considered about the chess piece color of the winner of the game, we are dropping outcomes that result to a draw.

b. **Label Encoding:** Label Encoder was used to transform categorical features, namely – WINNER, VICTORY STATUS, OPENING NAME, OPENING CATEGORY, GAME LEVEL, GAME VARIANT, MOVES

   The outcome variable WINNER was encoded as:

   0 = Black

   1 = White

c. **Choosing Features for Prediction:** The final list of chosen features used as predictor variables is: RATED, TURNS, VICTORY STATUS, INCREMENT CODE, WHITE RATING, BLACK RATING, OPENING NAME, OPENING PLY, MATCH DURATION, OPENING CATEGORY, MEAN RATING, GAME LEVEL, GAME VARIANT.

d. **Splitting data into training data and testing data:** The dataset was split into 60% training data and 40% testing data stratified on 'WINNER'.

e. **Normalizing and Scaling Data:** The data was normalized and scaled using methods from sklearn library.

VI. PREDICTION OUTCOME AND ANALYSIS

The outcome and analysis corresponding to each algorithm is detailed below:

1. **Naïve Bayes:** At first, we chose Gaussian Naive Bayes, which is a probabilistic algorithm used for classification tasks.

   Reasons for choosing Naïve Bayes for Winner prediction:

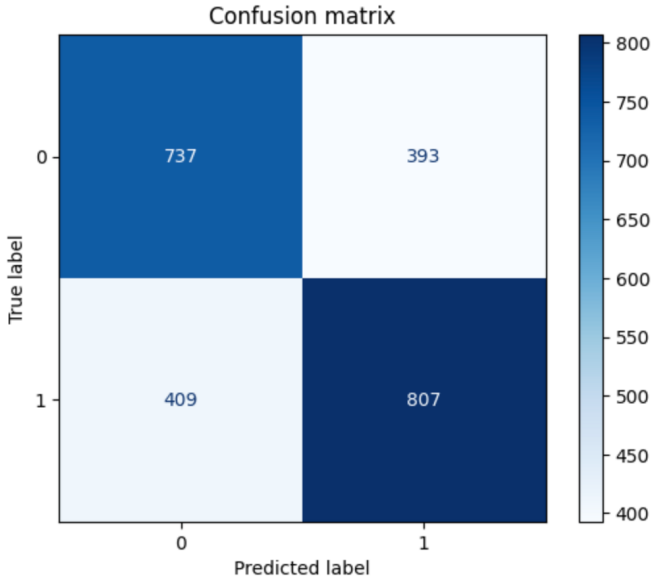
   a. **Handles both continuous and categorical data efficiently:** The Lichess Chess Game Dataset has diverse set of features such as ratings of players, moves made by each player etc. Since Gaussian Naïve Bayes can handle both continuous and categorical data efficiently, this was chosen as one of the models.

   b. **Reduces risk of overfitting for high-dimensional data:** The Lichess Chess Game Dataset contains high-dimensional data and since Naïve Bayes assumes that each feature is independent of the others, it reduces the risk of overfitting the data.

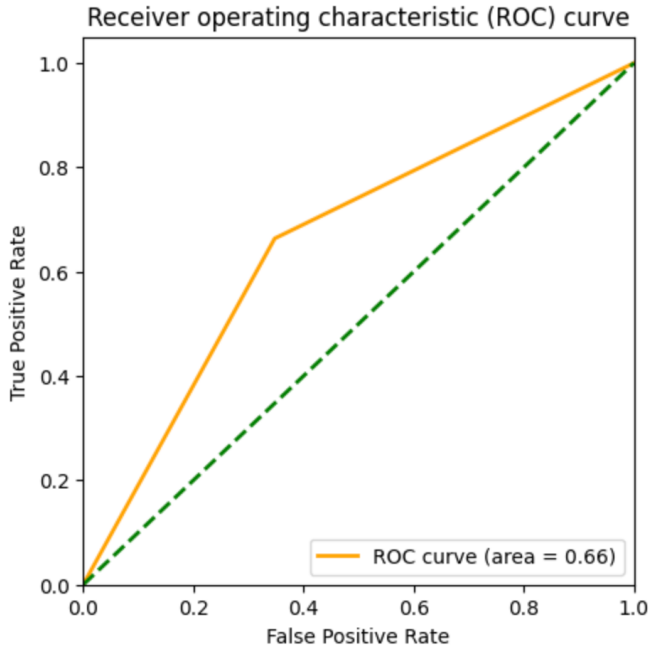
   Evaluation Metrics

```
Accuracy of the Model =  0.6581415174765558
Precision Score =  [0.64310646 0.6725    ]
F1 Score =  0.668046357615894
Recall =  0.6636513157894737

 Classification Report
              precision    recall  f1-score   support

           0       0.64      0.65      0.65      1130
           1       0.67      0.66      0.67      1216

    accuracy                           0.66      2346
   macro avg       0.66      0.66      0.66      2346
weighted avg       0.66      0.66      0.66      2346
```

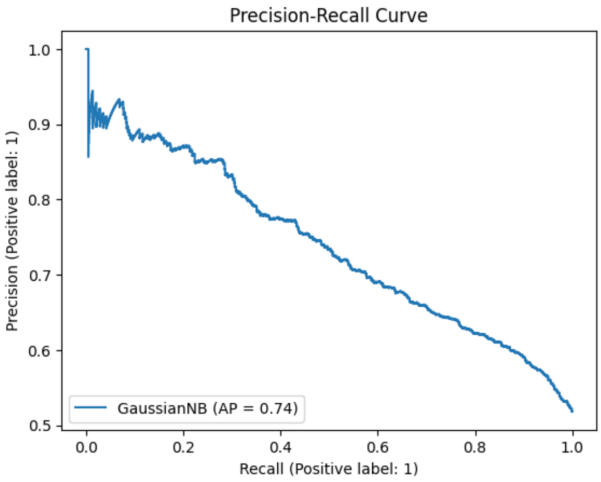
   Confusion Matrix: The model correctly classified 737 records as positive and 807 records as negative. The model incorrectly classified 409 records as negative and 807 records as positive. This model is correctly identifying a significant proportion of the positive cases while minimizing false positives. However, there is still room for improvement.



ROC Curve: This model is correctly identifying the positive cases 66% of the time and incorrectly identifies the negative cases as positive 34% of the time.



Precision Recall Curve: An Average Precision score of 0.74 signifies that the model is able to correctly identify a relatively high proportion of true positives while minimizing false positives.

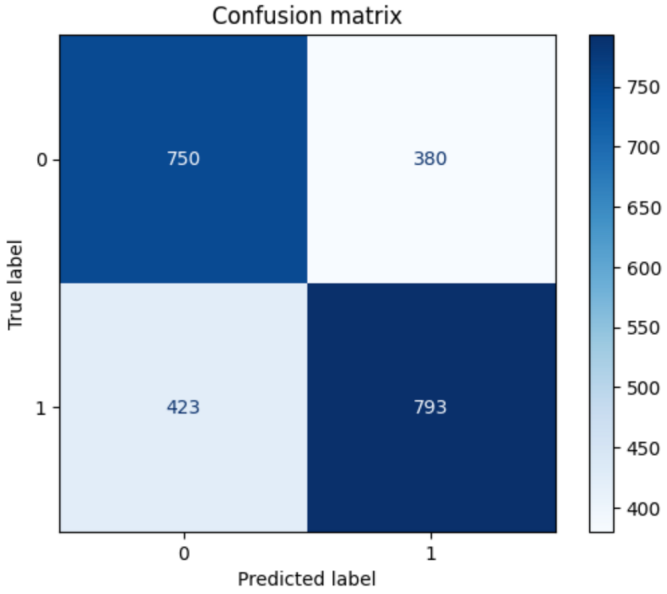


Improvement upon base model: To improve upon the accuracy of Gaussian Naïve Bayes, Bootstrap Aggregating (Bagging) was used with Naïve Bayes Classifier as the base model. Bagging helps in the reduction of variance by selecting different subsets of the training dataset with replacement and is particularly helpful with high-dimensional datasets such as the Lichess Chess Game Dataset. Results obtained by implementing bagging are as follows:

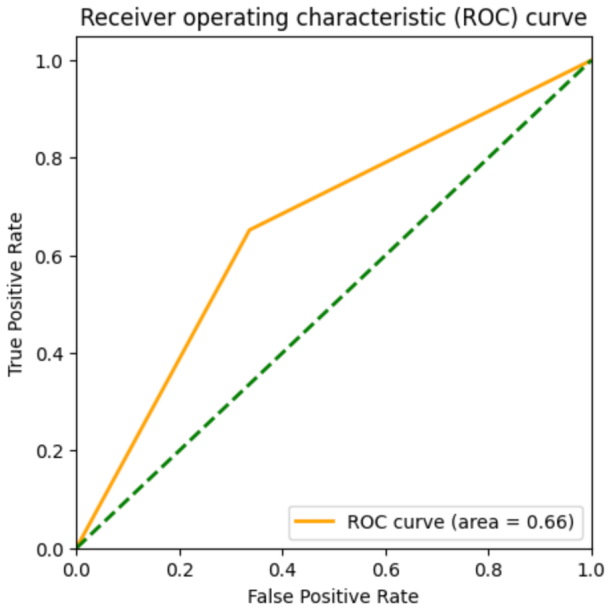
Evaluation Metrics

```
Accuracy of the Model =  0.6577152600170503
Precision Score =  [0.63938619 0.67604433]
F1 Score =  0.663876098786103
Recall =  0.6521381578947368

 Classification Report
              precision    recall  f1-score   support

           0       0.64      0.66      0.65      1130
           1       0.68      0.65      0.66      1216

    accuracy                           0.66      2346
   macro avg       0.66      0.66      0.66      2346
weighted avg       0.66      0.66      0.66      2346
```

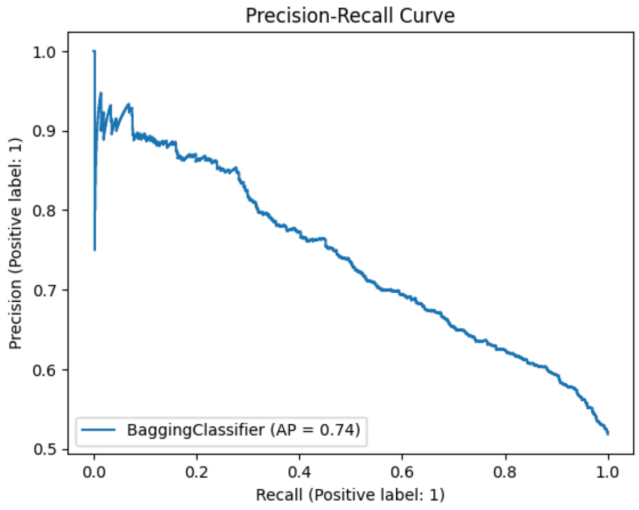
Confusion Matrix: The model correctly classified 750 records as positive and 793 records as negative. The model incorrectly classified 423 records as negative and 380 records as positive. False Negative count increased, and True Negative count decreased, this means that the model is becoming less accurate in identifying negative cases.



ROC Curve: This model is correctly identifying the positive cases 66% of the time and incorrectly identifies the negative cases as positive 34% of the time.



Precision Recall Curve: An Average Precision score of 0.74 signifies that the model is able to correctly identify a relatively high proportion of true positives while minimizing false positives.



Conclusion: The evaluation metrics in both the variants of the model performed nearly similar.

2. **K-Nearest Neighbor:** Next, we tried non-parametric, supervised learning classifier K-Nearest Neighbor for prediction.

   Reasons for choosing K-Nearest Neighbor for Winner prediction:

   a. K-Nearest Neighbo**r** is a non-parametric method, which means it makes no assumptions about the data distribution. As a result, it is particularly suited to datasets with complex relationships and interactions, such as Lichess Chess Game Dataset.

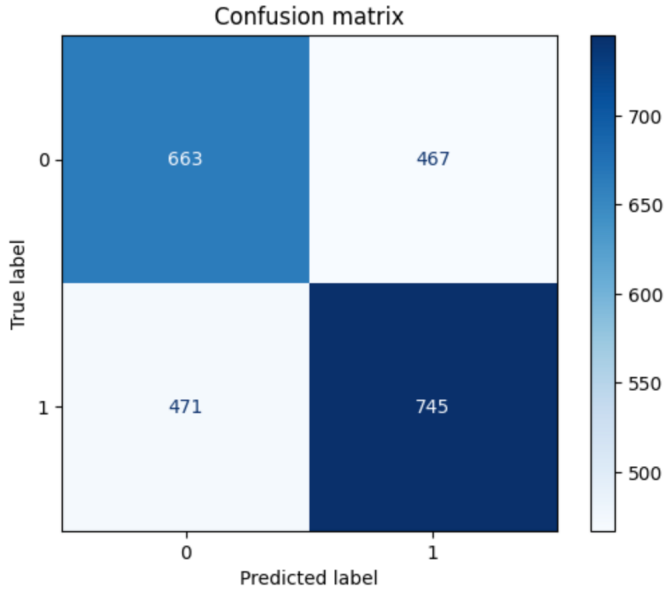
   b. When the number of neighbors considered (K value) is tuned for the dataset, K-Nearest Neighbor can achieve excellent accuracy on classification tasks.

   c. K-Nearest Neighbor can capture non-linear relationships between the features and the target variable. Liches Chess Game Dataset includes complex features like match duration, moves, opening category etc. because of which a linear decision boundary may not be able to accurately separate the classes. In such cases, non-linear algorithms like KNN can yield a better performance.

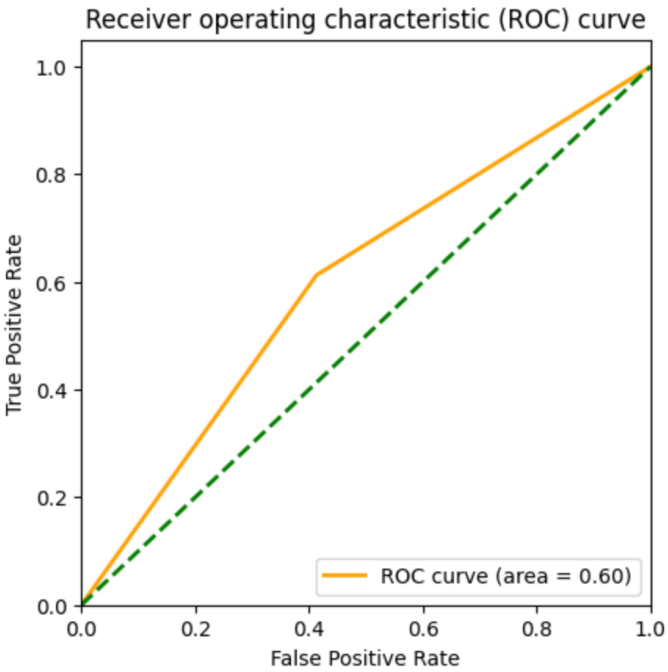
Evaluation Metrics

```
Accuracy of the Model =  0.6001705029838023
Precision Score =  [0.58465608 0.61468647]
F1 Score =  0.6136738056013179
Recall =  0.6126644736842105

 Classification Report
              precision    recall  f1-score   support

           0       0.58      0.59      0.59      1130
           1       0.61      0.61      0.61      1216

    accuracy                           0.60      2346
   macro avg       0.60      0.60      0.60      2346
weighted avg       0.60      0.60      0.60      2346
```

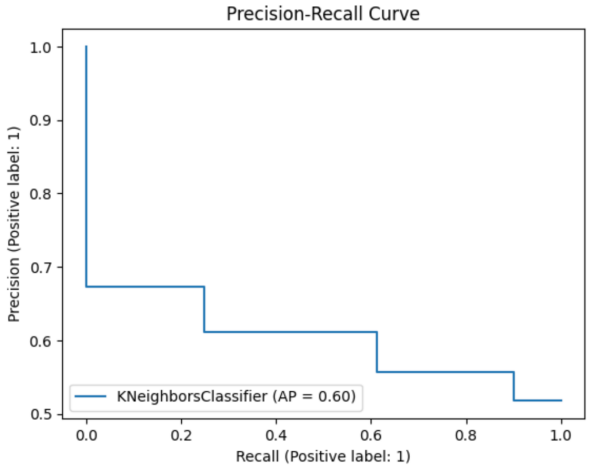
Confusion Matrix: The model correctly classified 637 records as positive and 745 records as negative. The model incorrectly classified 471 records as negative and 745 records as positive. False Positive and False Negative values are high which suggests that the model is less accurate overall and is making errors in its predictions.



ROC Curve: This model is correctly identifying the positive cases 60% of the time and incorrectly identifies the negative cases as positive 40% of the time.



Precision Recall Curve: An Average Precision score of 0.60 signifies that the model is performing moderately well. The model is correctly identifying a reasonable proportion of positive cases, but there are also some false positives and false negatives.

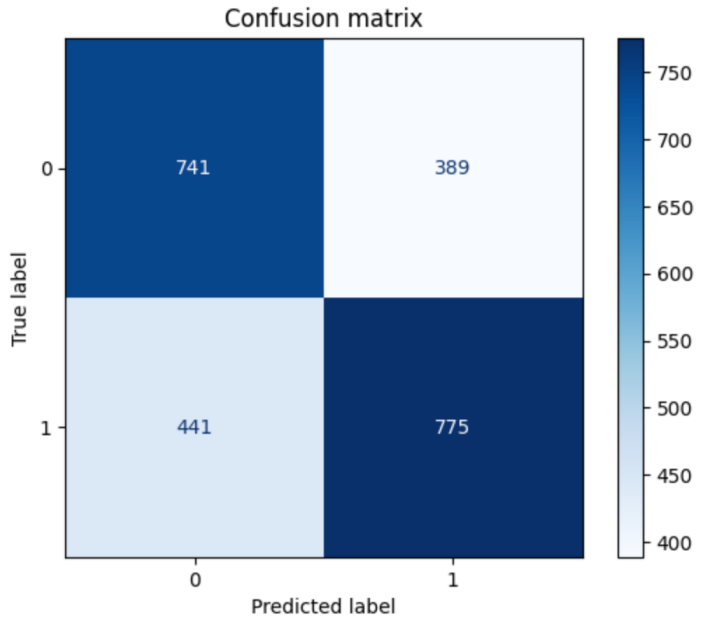


Improvement upon base model: To improve upon the accuracy of KNN classifier, we used GridSearchCV with KNN. GridSearchCV performs an exhaustive search over a specified parameter grid of 5, 15 and 20 nearest neighbors to find the optimal set of hyperparameters for the model. The results are as shown below:

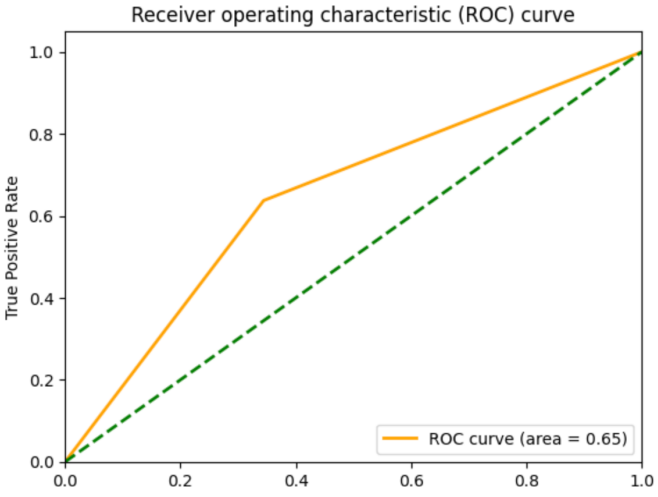
Evaluation Metrics

```
Accuracy of the Model =  0.6462063086104007
Precision Score =  [0.62690355 0.66580756]
F1 Score =  0.6512605042016806
Recall =  0.6373355263157895

 Classification Report
              precision    recall  f1-score   support

           0       0.63      0.66      0.64      1130
           1       0.67      0.64      0.65      1216

    accuracy                           0.65      2346
   macro avg       0.65      0.65      0.65      2346
weighted avg       0.65      0.65      0.65      2346
```

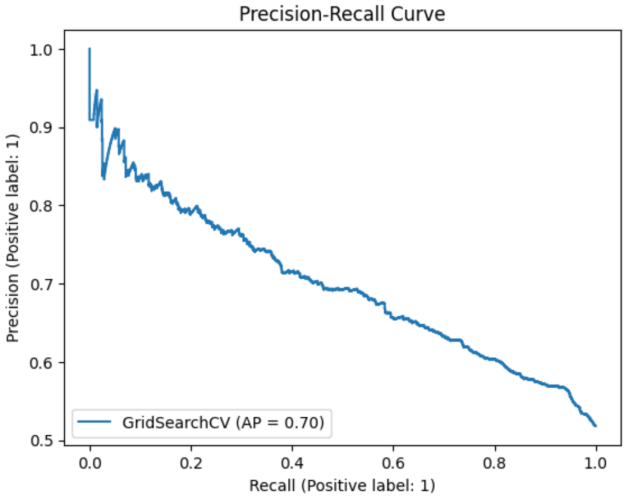
Confusion Matrix: The model correctly classified 741 records as positive and 775 records as negative. The model incorrectly classified 441 records as negative and 389 records as positive. True Positive and True Negative counts increased while False Positive and False Negative counts decreased which suggests that this model is better at overall prediction.



ROC Curve: This model is correctly identifying the positive cases 65% of the time and incorrectly identifies the negative cases as positive 35% of the time.



Precision Recall Curve: An Average Precision score of 0.70 indicates that the model is correctly identifying a reasonable proportion of positive cases, while keeping false positives to a minimum.



Conclusion: K-Nearest Neighbor with GridSearchCV showed an improved performance over all evaluation metrics compared to the base KNN model.

3. **Logistic Regression:** The next model we tried with the Liches Chess Game Dataset to evaluate the performance is Logistic Regression.

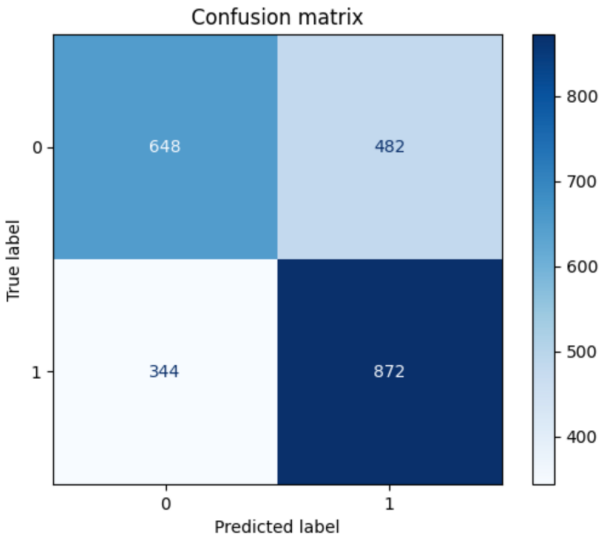
   Reasons for choosing Logistic Regression for Winner prediction:

   a. Logistic regression was designed specifically to model binary outcomes, making it helpful for predicting the winner of a chess game with only two potential outcomes – white or black.

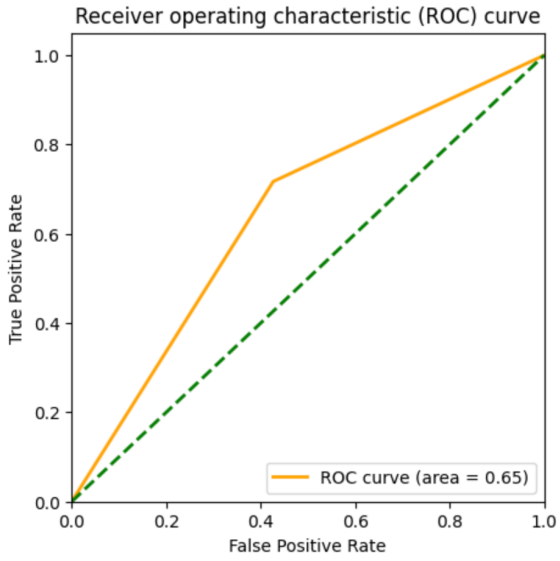
Evaluation Metrics

```
Accuracy of the Model =  0.6479113384484229
Precision Score =  [0.65322581 0.64401773]
F1 Score =  0.6785992217898832
Recall =  0.7171052631578947

 Classification Report
              precision    recall  f1-score   support

           0       0.65      0.57      0.61      1130
           1       0.64      0.72      0.68      1216

    accuracy                           0.65      2346
   macro avg       0.65      0.65      0.64      2346
weighted avg       0.65      0.65      0.65      2346
```

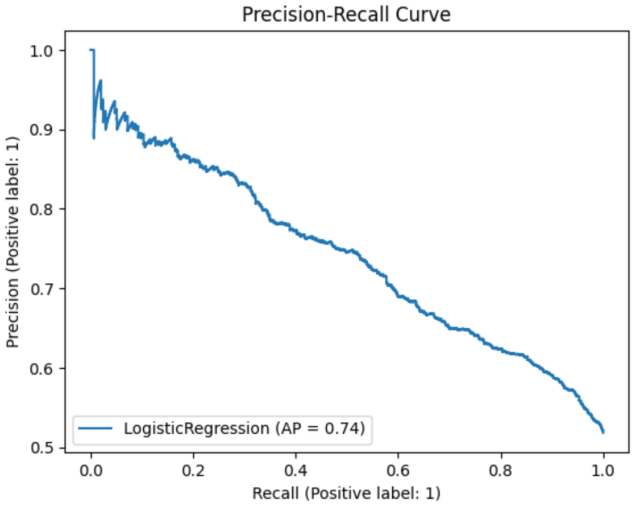
Confusion Matrix: The model correctly classified 648 records as positive and 872 records as negative. The model incorrectly classified 344 records as negative and 482 records as positive. False positive count is greater than False Negative which suggests that the model is sensitive to the positive class and may be incorrectly identifying negative cases as positive.



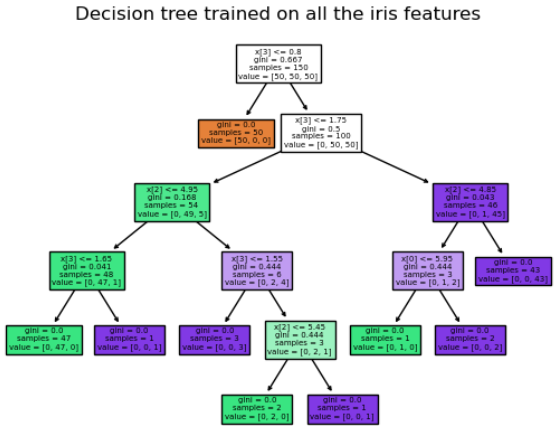
ROC Curve: This model is correctly identifying the positive cases 65% of the time and incorrectly identifies the negative cases as positive 35% of the time.



Precision Recall Curve: An Average Precision score of 0.74 signifies that the model is able to correctly identify a relatively high proportion of true positives while minimizing false positives.



4. **Decision Tree:** Decision Tree is a type of non-parametric supervised learning approach that is used for classification and regression problems. Decision Tree models create simple decision rules based on the available data and uses these rules to predict the value of a target variable. They can be thought of as a set of piecewise constant functions that approximate the target variable with the constant values corresponding to the expected values of the target variable within each decision boundary.
The figure below shows an example of Decision Tree.



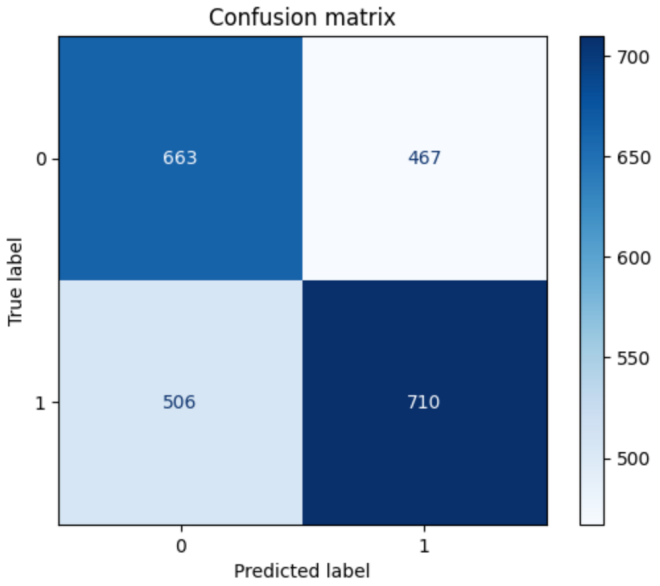
Reasons for choosing Decision Tree for Winner prediction:

a. Decision trees naturally perform feature selection by choosing the most essential features at each split. In chess, it is critical to determine which features are most significant for predicting the winner. These traits can be identified automatically using decision trees, resulting in improved performance.

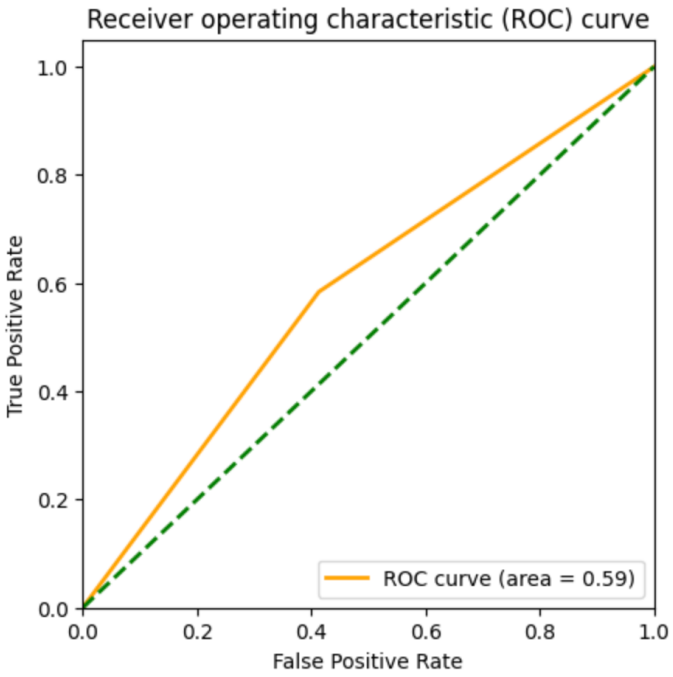
Evaluation Metrics

```
Accuracy of the Model =  0.5852514919011083
Precision Score =  [0.56715141 0.60322855]
F1 Score =  0.593397409109904
Recall =  0.5838815789473685

 Classification Report
              precision    recall  f1-score   support

           0       0.57      0.59      0.58      1130
           1       0.60      0.58      0.59      1216

    accuracy                           0.59      2346
   macro avg       0.59      0.59      0.59      2346
weighted avg       0.59      0.59      0.59      2346
```

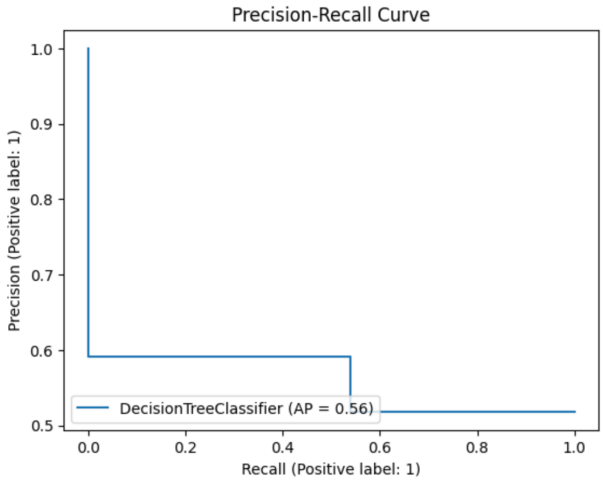
Confusion Matrix: The model correctly classified 663 records as positive and 710 records as negative. The model incorrectly classified 506 records as negative and 467 records as positive. False Negative count is high which suggests that this model is suggests that the model is not sensitive enough to the positive class and is failing to identify many positive cases.



ROC Curve: This model is correctly identifying the positive cases 59% of the time and incorrectly identifies the negative cases as positive 41% of the time.



Precision Recall Curve: An Average Precision score of 0.56 suggests that the model is correctly identifying some positive cases but may also be making some false positive and false negative predictions.

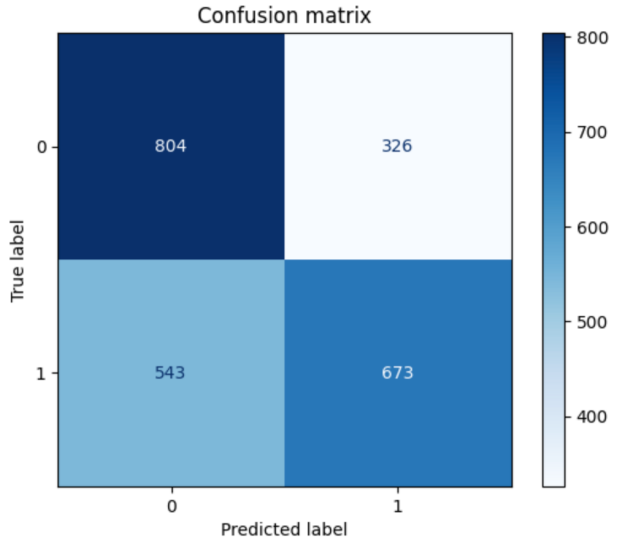


Improvement upon base model: To improve upon the accuracy of Decision Tree model, we used GridSearchCV with Decision Tree and explored the accuracy by varying the maximum depth of the decision tree. The results obtained are as follows:

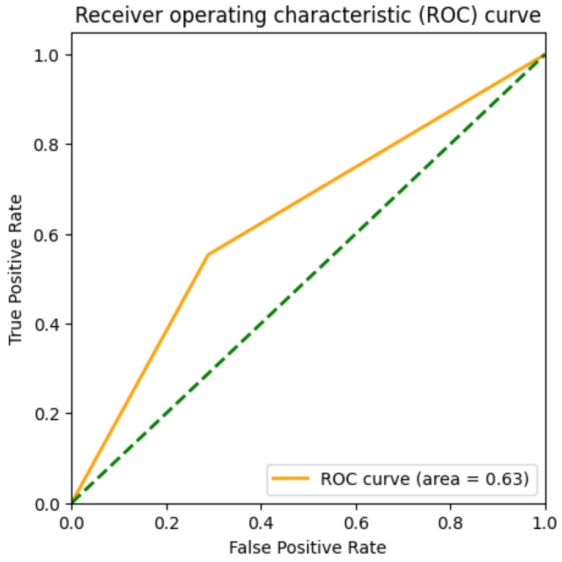
Evaluation Metrics

```
Accuracy of the Model =  0.6295822676896846
Precision Score =  [0.59688196 0.67367367]
F1 Score =  0.6076749435665915
Recall =  0.553453947368421

 Classification Report
              precision    recall  f1-score   support

           0       0.60      0.71      0.65      1130
           1       0.67      0.55      0.61      1216

    accuracy                           0.63      2346
   macro avg       0.64      0.63      0.63      2346
weighted avg       0.64      0.63      0.63      2346
```

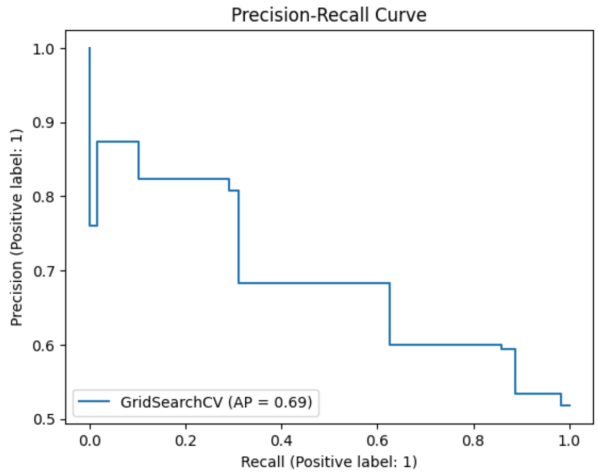
Confusion Matrix: The model correctly classified 804 records as positive and 673 records as negative. The model incorrectly classified 543 records as negative and 326 records as positive. False Negative increased while False Positive decreased which suggests that the model is in incorrectly classifying more positive cases as negative and is correctly identifying more negative cases.



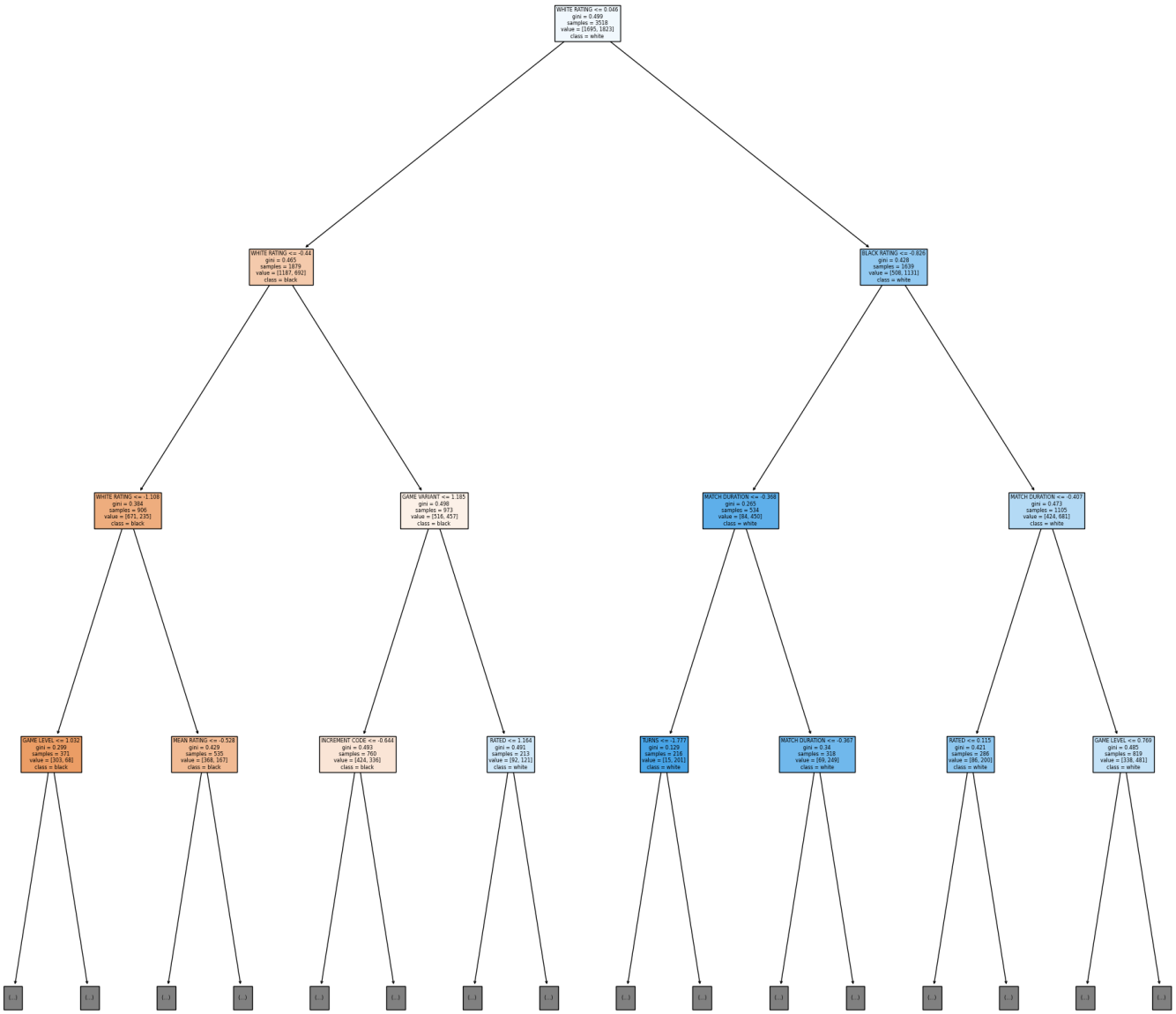
ROC Curve: This model is correctly identifying the positive cases 63% of the time and incorrectly identifies the negative cases as positive 37% of the time.



Precision Recall Curve: An Average Precision score of 0.56 suggests that the model is correctly identifying a high proportion of positive cases, with a relatively low number of false positives and false negatives.



Decision Tree Graph Corresponding to the Results Obtained



Conclusion: Decision Tree model with GridSearchCV performed better than the base Decision Tree model across all evaluation metrics.

5. **Random Forest:** A random forest is a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

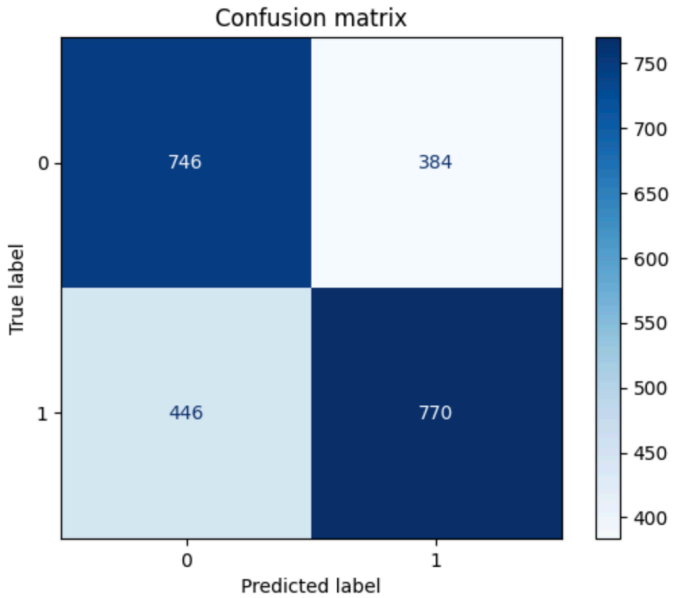
   Reasons for choosing Random Forest for Winner prediction:

   a. When compared to individual decision trees, random forest can enhance prediction accuracy. This is because random forest combines multiple decision trees, each trained on a different subset of data, to provide a more accurate prediction. Hence, we chose Random Forest as one of the models for classification.

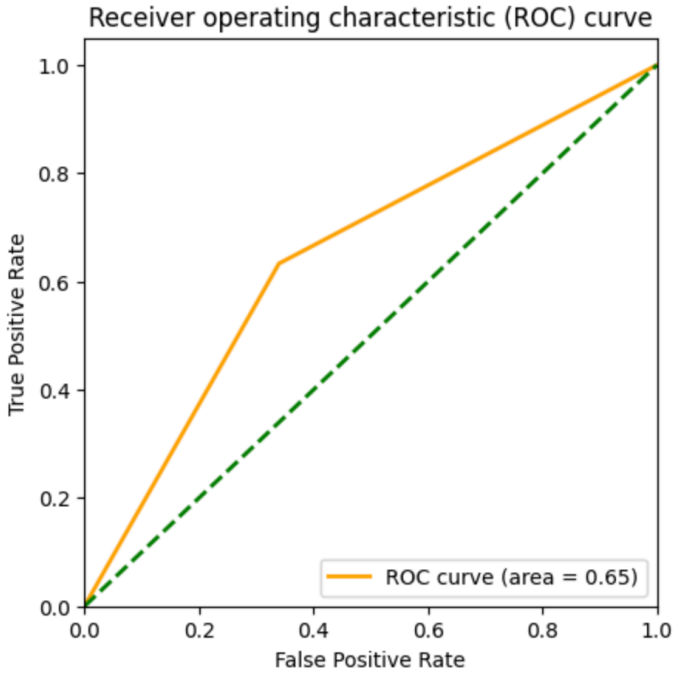
Evaluation Metrics

```
Accuracy of the Model =  0.6462063086104007
Precision Score =  [0.62583893 0.66724437]
F1 Score =  0.649789029535865
Recall =  0.6332236842105263

 Classification Report
              precision    recall  f1-score   support

           0       0.63      0.66      0.64      1130
           1       0.67      0.63      0.65      1216

    accuracy                           0.65      2346
   macro avg       0.65      0.65      0.65      2346
weighted avg       0.65      0.65      0.65      2346
```

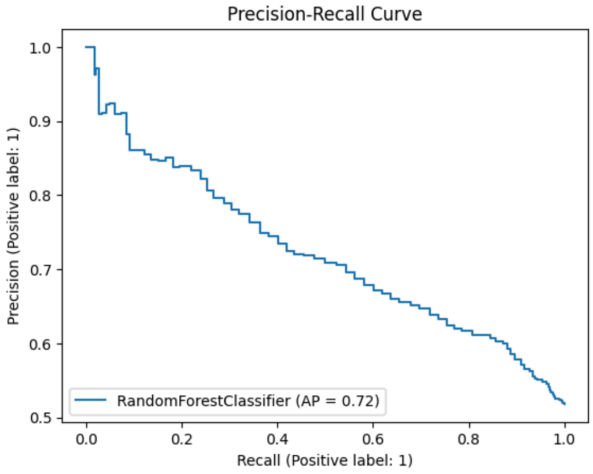
Confusion Matrix: The model correctly classified 746 records as positive and 770 records as negative. The model incorrectly classified 446 records as negative and 384 records as positive. False Negative is higher than False Positive which suggests that the model is failing to identify positive cases while correctly identifying most of the negative cases. The model may be setting a high threshold for positive predictions which may cause a misclassification of positive cases.



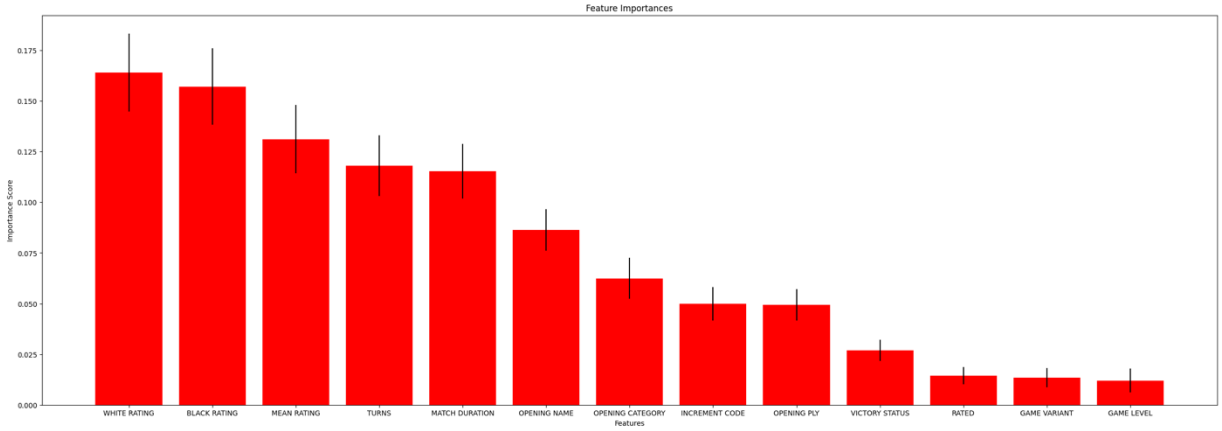
ROC Curve: This model is correctly identifying the positive cases 65% of the time and incorrectly identifies the negative cases as positive 35% of the time.



Precision Recall Curve: An Average Precision score of 0.72 signifies that the model is able to correctly identify a relatively high proportion of true positives while minimizing false positives.



Feature Importance Map for Random Forest Model: The importance of the features in the decision-making process is decreasing from WHITE RATING to GAME LEVEL with GAME LEVEL contributing the minimum and WHITE RATING contributing the maximum.



Conclusion: The Random Forest model performed better than the Decision Tree model and the importance score of each feature in the training process is as shown by the Feature Importance Map. White Rating was a heavily weighted feature in the decision-making process whereas Game Level had the least importance.

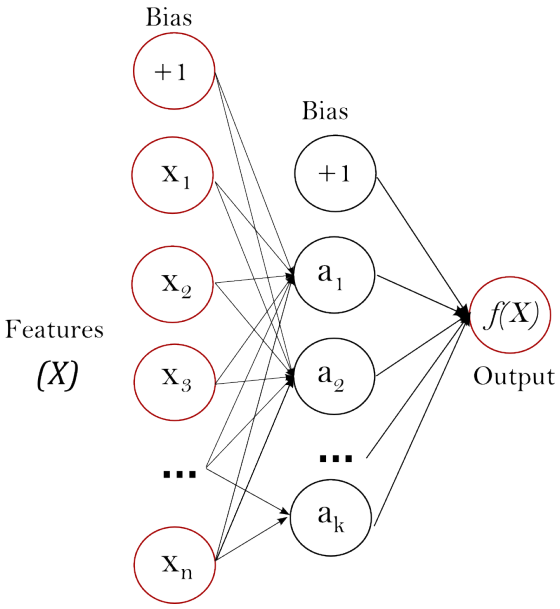
6. **Neural Network - Multi-Layer Perceptron:** Multi-Layer Perceptron is a supervised learning algorithm that learns a function $f(\cdot)$ by training on a given dataset.

   Function: $f(\cdot) = R^m \rightarrow R^o$

   m = Number of dimensions for input.

   o = Number of dimensions for output.

   The figure below shows one hidden layer Multi-Layer Perceptron with scalar output.



Reasons for choosing Multi-layer Perceptron for Winner prediction:

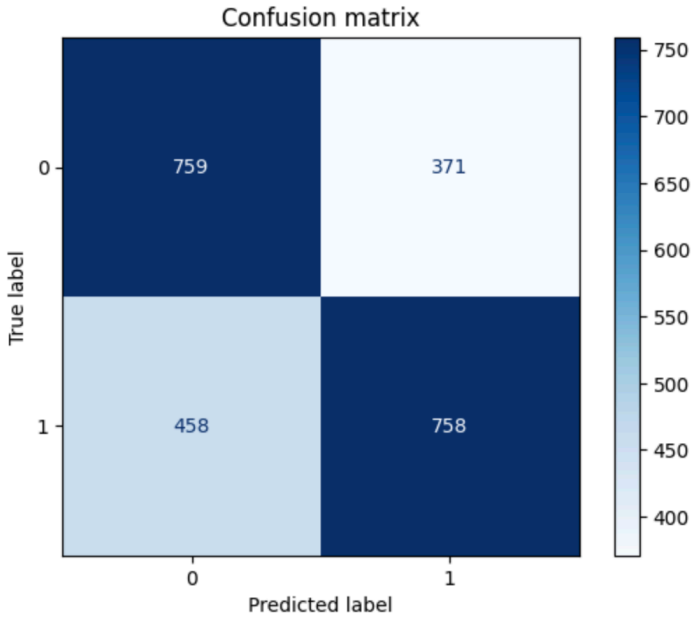
a. Multi-Layer Perceptron is a neural network and can generalize better on new and unseen data, making it suitable for the problem of winner prediction in a chess game, where the aim is to predict the outcome of future matches.

b. Multi-Layer Perceptron is capable of learning complicated, non-linear correlations between features, allowing them to make correct predictions and thereby achieving a high accuracy on test data.

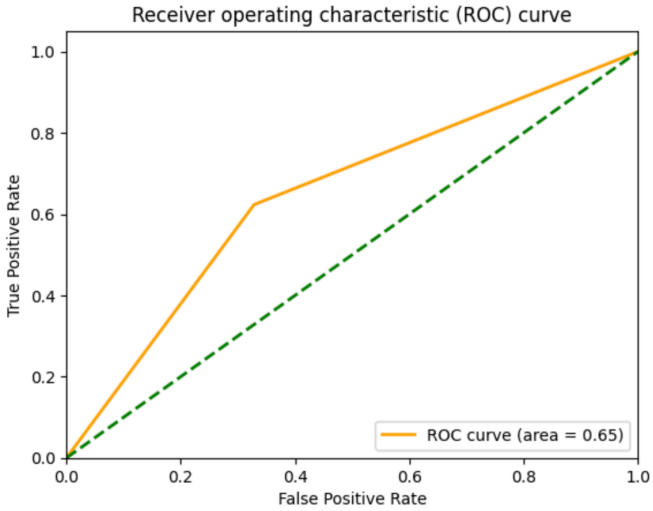
Evaluation Metrics

```
Accuracy of the Model =  0.6466325660699063
Precision Score =  [0.62366475 0.67139061]
F1 Score =  0.6464818763326226
Recall =  0.6233552631578947

 Classification Report
              precision    recall  f1-score   support

           0       0.62      0.67      0.65      1130
           1       0.67      0.62      0.65      1216

    accuracy                           0.65      2346
   macro avg       0.65      0.65      0.65      2346
weighted avg       0.65      0.65      0.65      2346
```

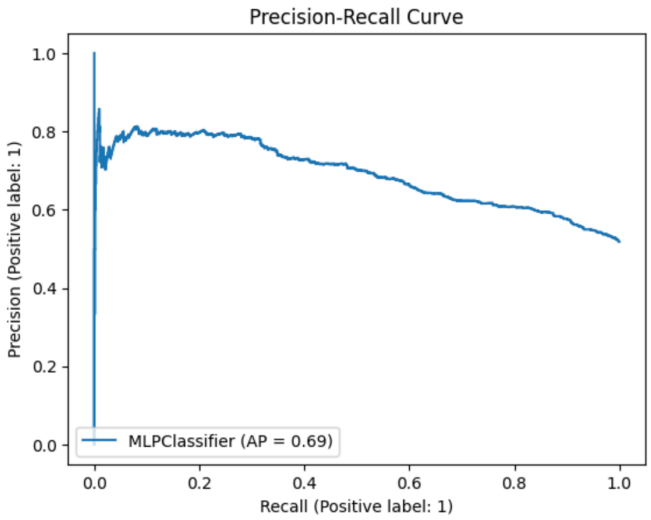
Confusion Matrix: The model correctly classified 759 records as positive and 758 records as negative. The model incorrectly classified 458 records as negative and 371 records as positive. False Negative is higher than False Positive which suggests that the model is failing to identify positive cases while correctly identifying most of the negative cases. The model may be setting a high threshold for positive predictions which may cause a misclassification of positive cases.



ROC Curve: This model is correctly identifying the positive cases 65% of the time and incorrectly identifies the negative cases as positive 35% of the time.



Precision Recall Curve: An Average Precision score of 0.69 signifies that the model is correctly identifying a high proportion of positive cases, with a relatively low number of false positives and false negatives.

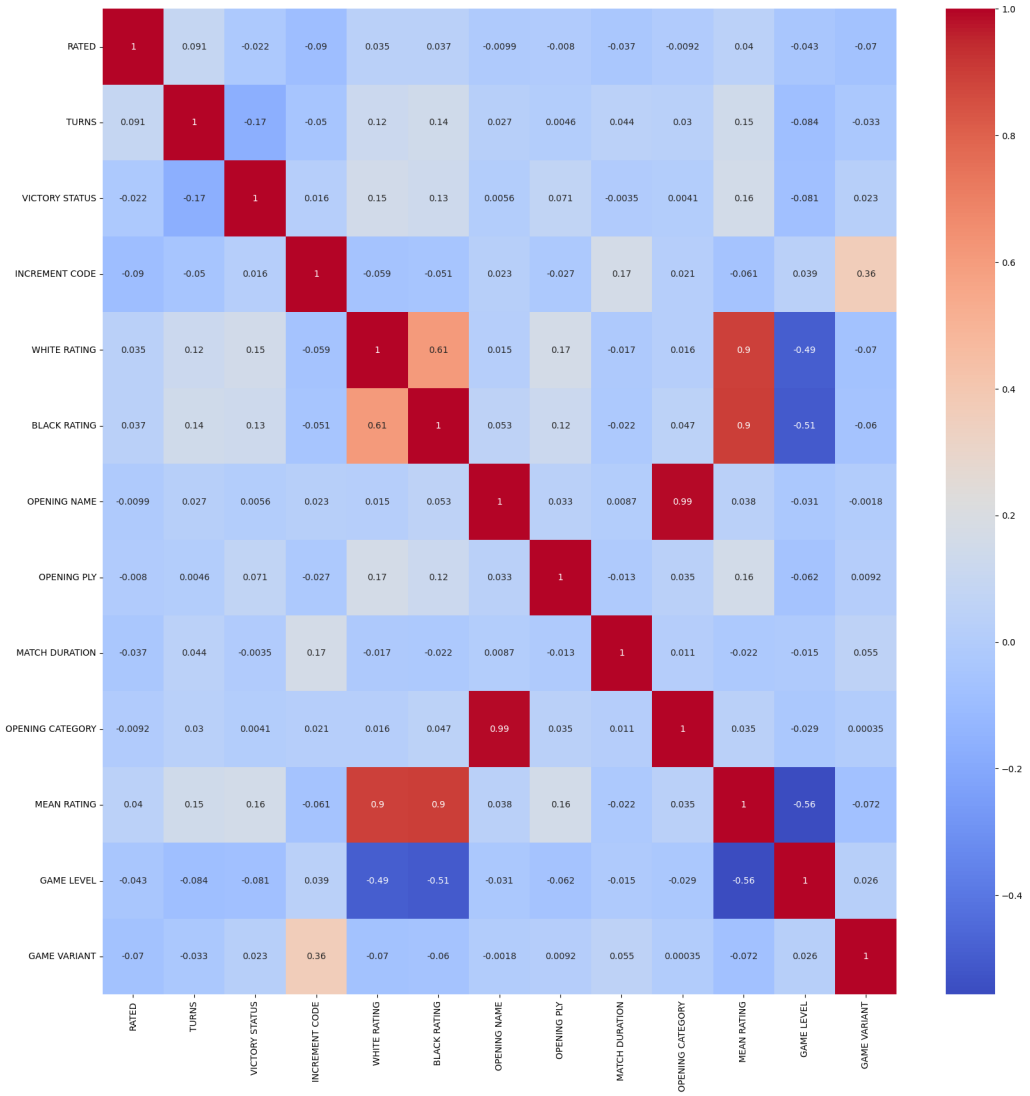


Conclusion: The accuracy obtained for Multi-Layer Perceptron was comparable to the other models used prior to this.

VII. CONCLUSION

From Phase 2 of the project, the following can be concluded.

a. All models used performed similar on the Lichess Chess Game Dataset and had comparable accuracy when predicting the winner of a chess game.
b. K-Nearest Neighbor and Decision Tree models integrating GridSearchCV with the base model gave better results.
c. The similarity across the performance, as measured through the evaluation metric, over all models used can be a result of the correlation between the features of the dataset. The co-relation between the features of the dataset us depicted in the co-relation matrix below.



ACKNOWLEDGMENT

We would like to take advantage of this opportunity to express our gratitude to Dr. Eric Mikida and Dr. Shamsad Parvin, our lecturers and Smarana Shrikant Pankati, our assigned project mentor for their insightful comments, patience, and time.

REFERENCES

1. https://bookdown.org/pq2142/finalproj/
2. https://community.ibm.com/community/user/ai-datascience/blogs/moloy-de1/2020/08/27/points-to-ponder
3. https://towardsdatascience.com/analysing-lichess-games-with-r-c4f8b0bc512c
4. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
5. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
6. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
7. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

8. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

9. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

10. https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html - :~:text=The precision-recall curve shows,a low false negative rate.