Final Project

Brandon Brodzinski

November 2022

Contents

1 Introduction

Chess is a board game that is one of the most popular games in the world. The game is composed of many patterns which offers plenty of options in the field of data mining and machine learning. It is said that there are more possible variations of chess games than there are atoms in the observable universe. In the game of chess there are white pieces and black pieces, the statistic shows that on average 55% of the time white will win because they have a first move advantage. For my final project, I decided to do a comparison of three supervised models to predict the winner of a chess game given the first 5 moves and effectively evaluate how accurate each algorithm is. The goal of this project is not to predict the best move, but to foretell the correct outcome of a match by using the probabilities of winning with respect to black. I plan to solve this issue utilizing divergent models and other classification strategies. The algorithms I will be using to compare are K-nearest Neighbor, AdaBoost, and Support Vector Classification.

2 Background of Models

2.1 KNN

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label in the case of classification[1]. It's important to choose an odd K value so that ties within classification are avoided. For example, if the K value is 5 and 2 labels are the digit 0, and 3 labels are the digit 8, then 8 would be the prediction since 3 is greater than 2. The pros of this algorithm are, no training period is required, it can be used for classification or regression, and a variety of distance criteria can be used. The cons are, decelerated performance in large datsets, does not adequately perform with high dimensions, and is sensitive to noisy data. Briefly, this algorithm solves classification problems easily as there are only two parameters, a K value and a distance.

2.2 AdaBoost

AdaBoost is an ensemble learning method (also known as "meta-learning") which was initially created to increase the efficiency of binary classifiers[2]. AdaBoost uses an iterative approach to learn from the mistakes of weak classifiers, and turn them into strong ones[2]. It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners[3]. In simple words, weak learners are converted into strong ones. The AdaBoost algorithm works on the same principle as boosting with a slight difference[3].

2.3 SVM

A support vector machine is a machine learning model that is able to generalise between two different classes if the set of labelled data is provided in the training set to the algorithm[4]. The main function of the SVM is to check for that hyperplane that is able to distinguish between the two classes[4]. There can be many hyper planes that can do this task but the objective is to find that hyperplane that has the highest margin that means maximum distances between the two classes, so that in future if a new data point comes that is two be classified then it can be classified easily[4].

3 Implementation

3.1 Preparing the Data

The data set I have chosen to utilize comes from `https://www.kaggle.com/datasets/datasnaek/chess` and is composed of 20,000 games that were played by various users on Lichess.org. This data set is composed of substantial information like openings, time increment, number of turns, and many more. I then pushed the data set to a GitHub repository `https://github.com/brodzy/chess-predictor/blob/main/games.csv` and converted the CSV to a raw file so that I could generate the data set in Google Colab.

3.2 Decision Points

I processed the data by first exterminating any columns that were not needed. Next, I split the "moves" column into indices and parsed the first 5 moves of each game. A "move" is completed once both White and Black have played one turn therefore 10 indices represents 5 moves. Next, I will calculate the probability of winning based on those set of moves as well as each opening with respect to black using the crosstab function in the pandas library. After I calculated the probabilities, I then created a dictionary and mapped the normalized numbers to each move and opening name. Additionally, I converted the "winner" column to binary so that I could standardize the data and use it correctly. Conclusively, I split my data into a training and testing split (80% / 20%) and evaluated each model using the classifications methods as mentioned via sklearn.

4 Evaluation

In this section, I review the results. In Figure 1, I created a confusion matrix of all three models to display the summary of what was classified correctly and incorrectly. All three matrices represented similar results in how they classified the test set with marginal differences in each sector of the matrix. In Figure 2, I created a heat map to compare the classification scores of the each algorithm. The results show that the SVC model did the best with 79% accuracy, AdaBoost in second with 77%, and KNN doing the worst with 75% accuracy. In Figure 3, I created a ROC (Receiver Operating Characteristic) graph to demonstrate the performance of all three models. This presents the trade-off between sensitivity (TPR) and specificity (FPR). The curve that is closest to the top left hand corner yields a better performance which in this case is SVC.

4.1 Confusion Matrices

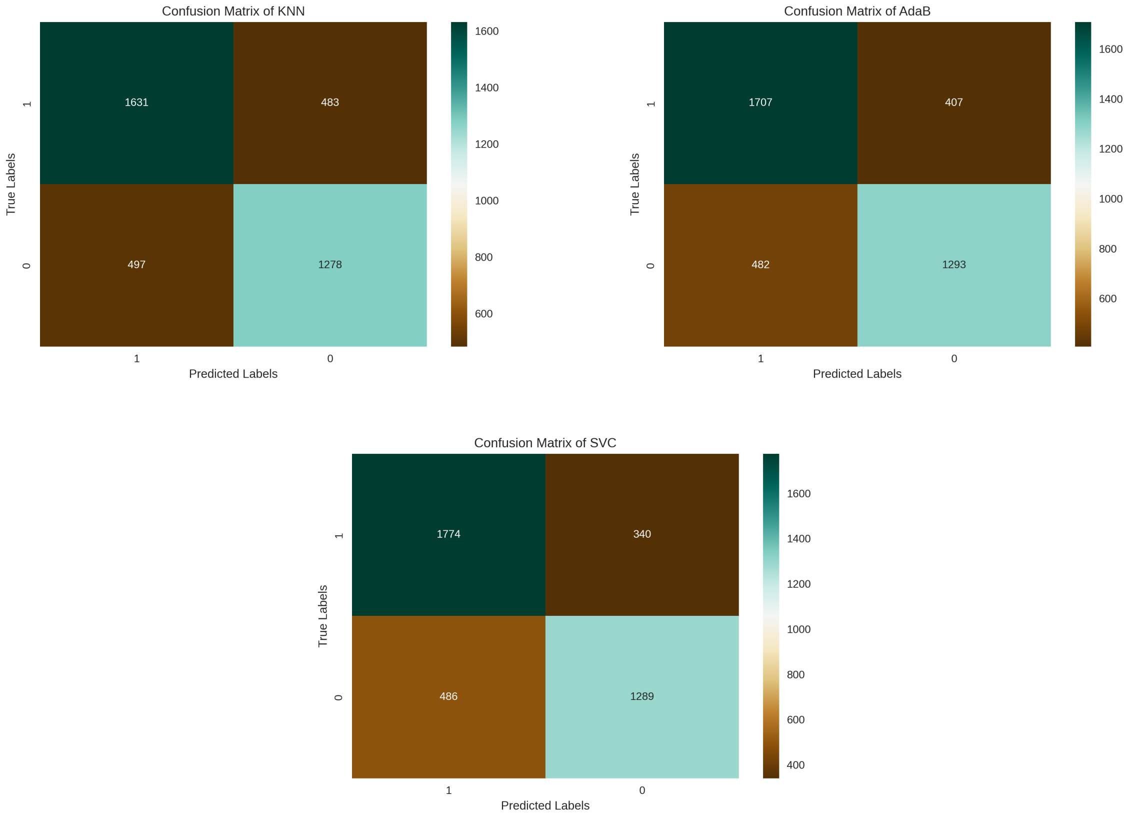

Figure 1: Confusion matrix prediction results.

4.2 Heat Map

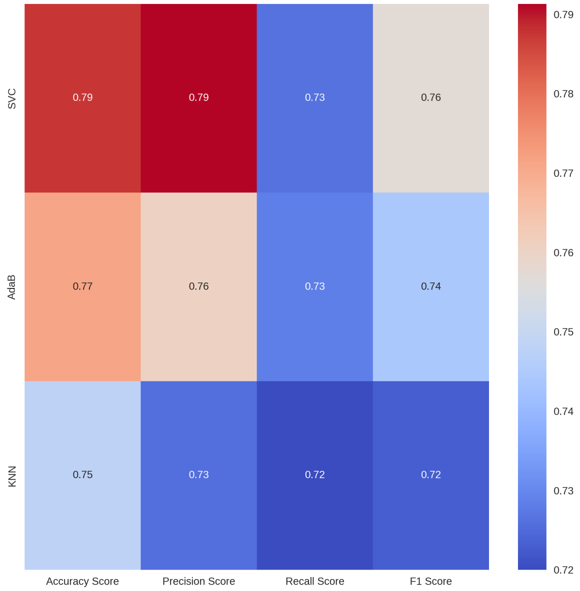

Figure 2: Model comparison via classification scores.

4.3 ROC Graph

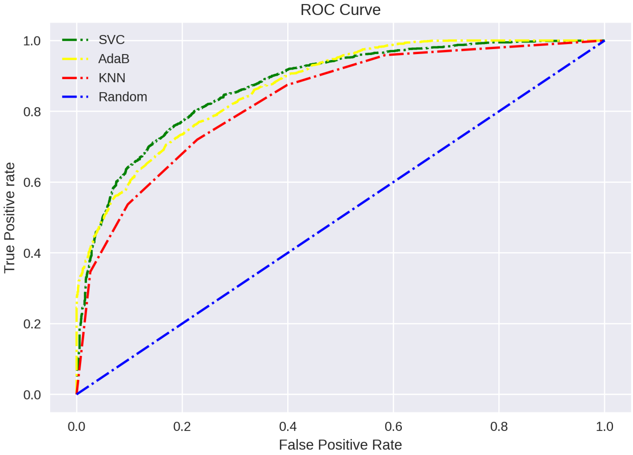

Figure 3: Accuracy performance based on classification model.

5 What I've learned

What I learned throughout this project is that some classification algorithms are better on different types of data sets. In my case, Support Vector Machine was the best algorithm used on the data set I chose resolving in a high of 81% accuracy after multiple test runs. I now have a comprehensive understanding of how a Support Vector Machine works by learning that it solves the complex optimization problem that maximizes its margin where the constraints say that points of each category should be on the correct side of the hyper plane. The pros of this algorithm are that it is easy to implement, interpret, and effective on a small size of training data. An alternative factor I discovered was the usefulness of a ROC curve to demonstrate performances of different models. It indicates the comparison of the true positive rate (y-axis) and false positive rate (x-axis) based on different thresholds. ROC does not depend on the class distribution which makes it useful for evaluating classifiers predicting rare events.

6 Conclusion

Chess is important because it teaches us about strategy, risk, and consequences similar to how life can be in the real world. Finding out the winner of a game is not only an alluring topic to explore but, can demonstrate practical data of statistics for players, openings, and time controls. This is considered a binary or ternary, (if counting draws) classification problem. Resolving an issue of this nature will disclose us more about the worth of pieces and different types of positions. I thoroughly enjoyed doing this topic because it is interesting to see how a game you love can translate to doing research in machine learning using different classification methods.

References

[1] https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 **Accessed: 2022-11-27**

[2] https://blog.paperspace.com/adaboost-optimizer/ **Accessed: 2022-11-27**

[3] https://www.mygreatlearning.com/blog/adaboost-algorithm/ **Accessed: 2022-11-27**

[4] https://www.analyticssteps.com/blogs/how-does-support-vector-machine-algorithm-works-machine-learning **Accessed: 2022-11-27**