**Wait, How Many Blunders?**

**An Analysis of Lichess Chess Games Statistics by**

Molly Rovinski, mrovinski@bellarmine.edu

Aaron Bone, abone@bellarmine.edu

**ABSTRACT**

This project is an exploratory analysis of a Chess game dataset provided by Lichess.com, one of the world’s most popular chess matchmaking sites. The games can range from beginner to grandmaster levels, contain various game types showing a wide variety of openings, and their miscalculations. The exploration portion of this report highlights varying correlations from variables like ‘total moves’, ‘long moves’, ‘inaccuracies’, ‘blunders’ (extremely bad moves that possibly flip the predicted winner), and ultimately the score. The Machine Learning portion of this report uses K Nearest Neighbor to properly estimate the winner of the game based solely off of these statistics without seeing the board at all.

**INTRODUCTION**

The purpose of this project is to preform both an exploratory and machine learning analysis on the Chess game dataset that is maintained by the Lichess database, which is one of the world’s most popular websites for online chess matches. The dataset that has been pulled for the purpose of this project ranges from September 1st through September 9th of 2020. Due to limitations in computer capacity, the dataset has been cut down to only contains games from September 1st. The games stored in this dataset include those ranging in difficulty, opening, movements, and calculations. The purpose of this project is to use the K-nearest neighbor to determine whether or not the Black will result in a win or not based entirely on the statistics of movements, game flips, and miscalculations.

**BACKGROUND**

1. **Data Set Description**

This data analysis evaluates the chess games played on Lichess.org from September 2020 that are collected and maintained by the Lichess database**.** The datasetdocuments chess games played between September 1, 2020. The original dataset, which includes games played between September 1st and September 9th, can be found at the following site: <https://www.kaggle.com/noobiedatascientist/lichess-september-2020-data?select=Sept_20_analysis.csv>

1. **General Jargon:**

* ELO: Rating system for zero sum games
* Opening: Opening sequence of moves, usually only one or two responses is enough to classify.
* ECO: Chess opening identifier acronyms, more detailed than an opening, it references a specific structure inside of an opening.
* Termination: How the game ended, either by checkmate, time, or resignation.
* Time Control: There are many formats of chess games. In general, there are 3 categories, bullet (60s), blitz, (180s), and rapid (10 Mins). These are generalizations, the column gives the time control in seconds.
* Increment: In some time controls, there’s a +3 seconds or +5 seconds, or +10 seconds, a minimum bonus time.
* Game Type: bullet, blitz, or rapid categorization.
* Blunder: Bad move that could have possibly flipped the entire engine evaluation of the game. A blunder is determined by Lichess’s game evaluation engine, Stockfish.
* Mistake: Not necessarily game changing, but harmful to the current evaluation.
* Inaccuracy: A move that slightly affects the evaluation bar.
* Game Flips: A moment where the evaluation flips from winner to loser.
* TS: Time Scramble, a moment where at least one player is close to losing on time.

1. **Machine Learning Model**

For prediction, we chose the K-Nearest Neighbor Machine Learning module. This module searches in a radius for nearby points, counting and comparing what the experimental point is closest to in correlation, then it makes a guess based on the location’s proximity. With how many variables we have, and how many decisive variables we have such as game flips and blunders, K Nearest Neighbor is a good fit. If an example point shares a significant number of blunders with a low total number of moves, it’s reasonable for a human to decide who won based off of that information alone, but a computer could use this sort of data to an even greater degree of analysis.

**EXPLORATORY ANALYSIS**

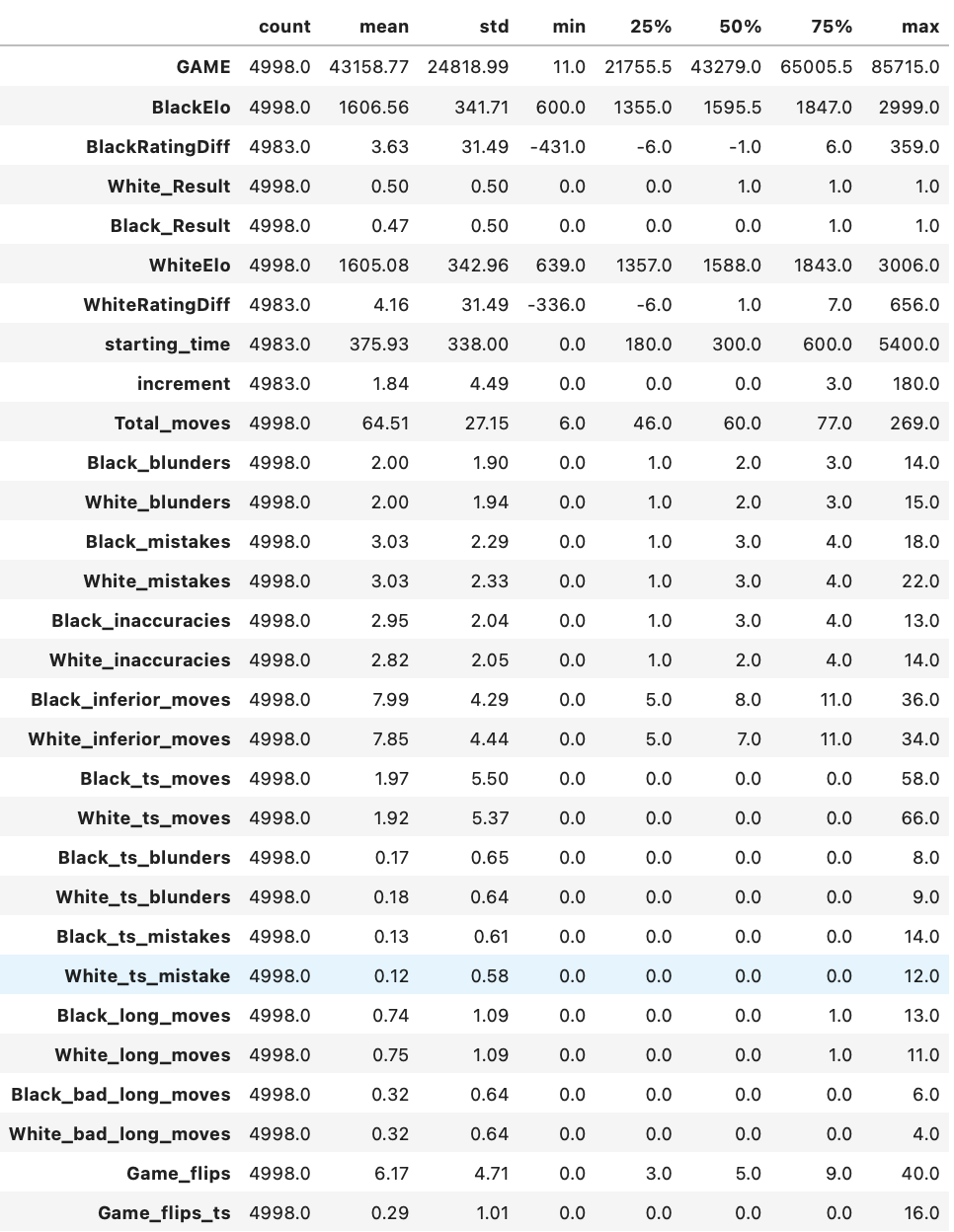
1. **Dataset Description**

This dataset contains 4998games with 40 columns of various data types. Though the original dataset included millions of games, the file had to be condensed to the first 4,998 for the purpose of running it. Additionally, two columns were added to the CSV file prior to beginning the exploratory analysis. Due to Excel automatically converting all 1-0 results into either a date of 1/1/2000 or an extended number of 36526, the columns had to be separated into ‘White\_Result’ and ‘Black\_Result’ through Excel functions. The complete list of variables and their types can be found in **Table 1**. Later on in the machine learning portion, a one other variable was created, but for the exploratory analysis it isn’t used. This columns are ‘Rating Difference’.None of the variables had a significant amount of missing data, with the highest being only 0.37% missing

**Table 1: Data Types**

|  |  |
| --- | --- |
| **Variable Name** | **Data Type** |
| Game | int64 |
| BlackElo | int64 |
| BlackRatingDiff | float64 |
| Date | object |
| ECO | object |
| White\_Result | float64 |
| Black\_Result | float64 |
| Opening | object |
| Termination | object |
| TimeControl | object |
| UTCTime | object |
| WhiteElo | int64 |
| WhiteRatingDiff | float64 |
| Black\_elo\_category | object |
| White\_elo\_category | object |
| starting\_time | float64 |
| increment | float64 |
| Game\_type | object |
| Total\_moves | int64 |
| Black\_blunders | int64 |
| White\_blunders | int64 |
| Black\_mistakes | int64 |
| White\_mistakes | int64 |
| Black\_inaccuracies | int64 |
| White\_inaccuracies | int64 |
| Black\_inferior\_moves | int64 |
| White\_inferior\_moves | int64 |
| Black\_ts\_moves | int64 |
| White\_ts\_moves | int64 |
| Black\_ts\_blunders | int64 |
| White\_ts\_blunders | int64 |
| Black\_ts\_mistakes | int64 |
| White\_ts\_mistakes | int64 |
| Black\_long\_moves | int64 |
| White\_long\_moves | int64 |
| Black\_bad\_long\_moves | int64 |
| White\_bad\_long\_moves | int64 |
| Game\_flips | int64 |
| Game\_flips\_ts | int64 |

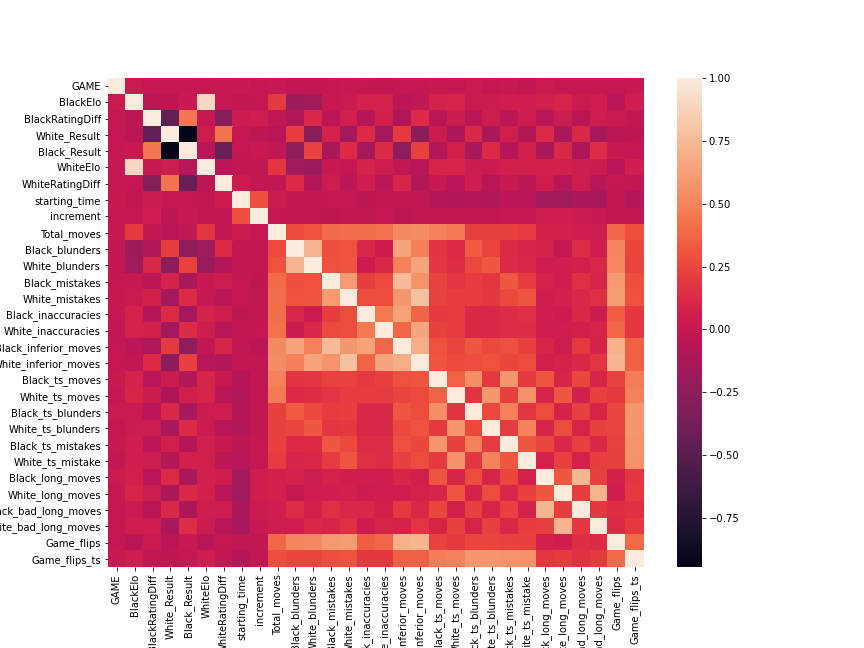
**Table 2: Summary Table**



1. **Correlations**

The following section contains the correlation matrix and heatmap for the quantitative variables within the dataset. This reflects the relationship between each continuous variable, with a visual heatmap of the correlation table following in **Figure 1.**  As can be seen in the heatmap, there are high correlations between the variables WhiteElo and BlackElo, while there is practically zero correlation between White\_Result and Black\_Result, which makes sense considering they would be the exact opposites of each other.

**Figure 1: Correlation Heat Map**

****

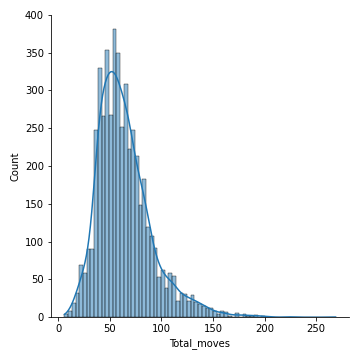
1. **Dataset Graphical Exploration**

The following section contains graphical representations of those variables within the dataset which seemed to have unusual distributions. These representations include distributions, scatterplots, and bar charts**.** Throughout the analysis, various variables had to be dropped due to their specificity to each chess game. Each will be explained within their own sections.

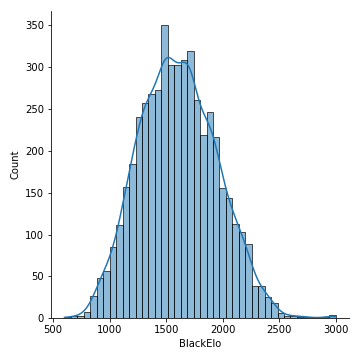
1. **Distributions**

There were a few variables that contained skewed distributions, which can be seen in the various graphs below. Most notably, the Total\_moves variable is shown to be skewed to the right with most of the outcomes centering around 50 moves, as can be seen in **Figure 2**. Additionally, two distributions that were of note were the BlackElo and WhiteElo variables, as they essentially had identical distributions, as can be seen in **Figures 3 and 4.** Finally, because White\_Result and Black\_Result were our added columns that will play a vital role in the machine learning portion, there distributions have been included in **Figures 5 and 6.**

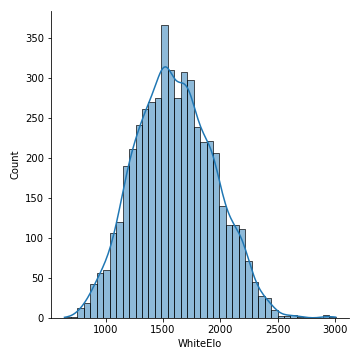
**Figure 2**

****

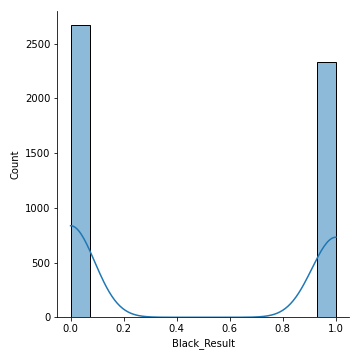
**Figure 3**

****

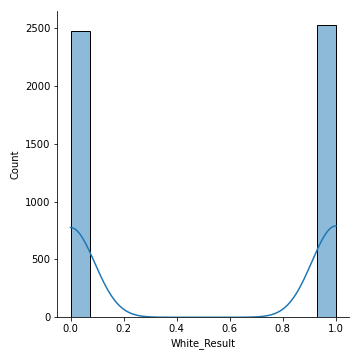
**Figure 4**

****

**Figure 5**

****

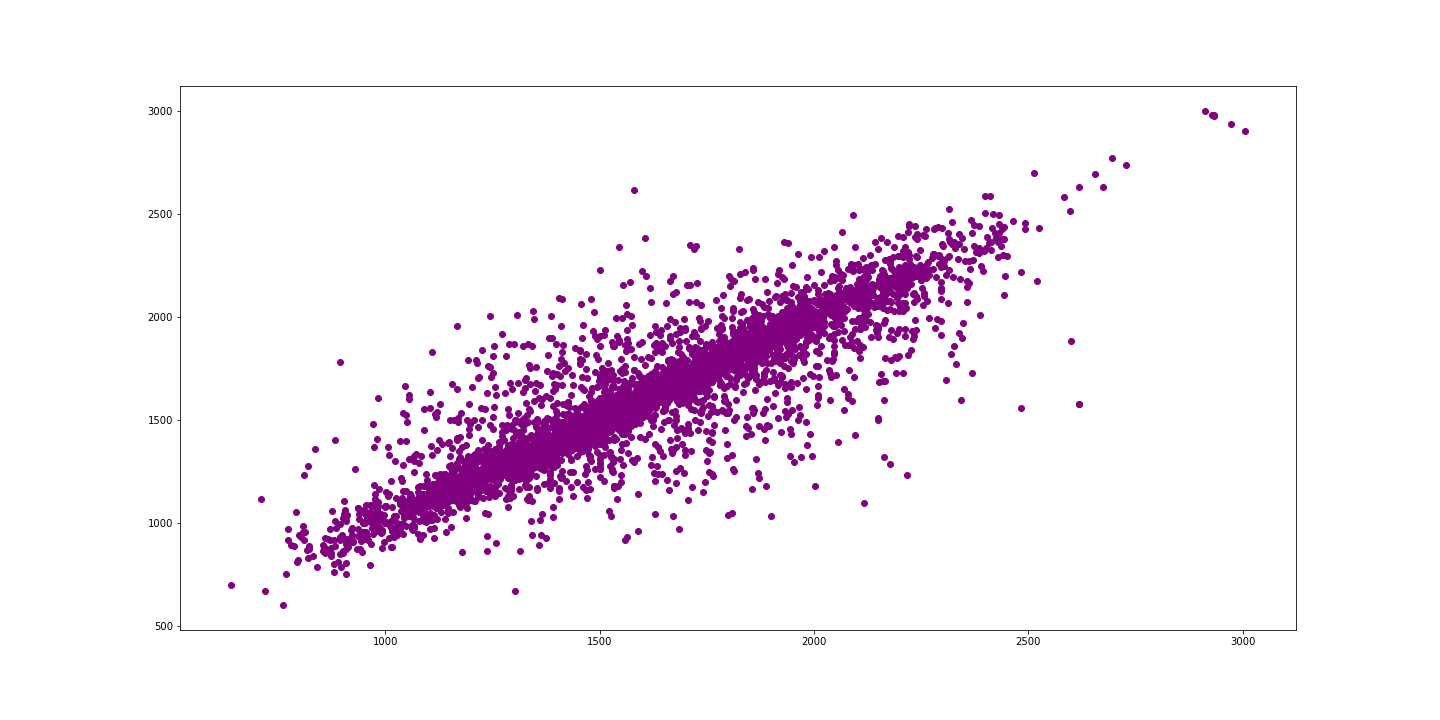
**Figure 6**

****

1. **Scatterplots**

To echo the interesting results seen in both the correlation heatmap and the distributions of WhiteElo and BlackElo, the Scatterplot comparing the two variables against each other emphasizes the close relationship that the two variables have with one another. This relationship can be seen in **Figure 7.**

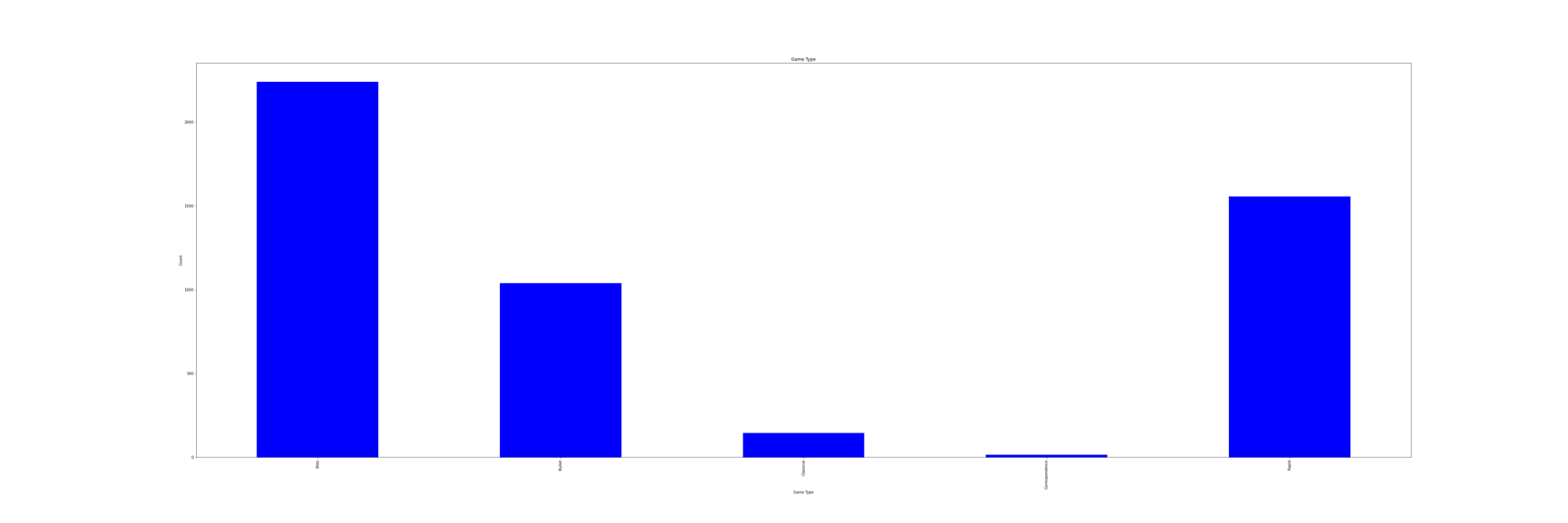
**Figure 7**

****

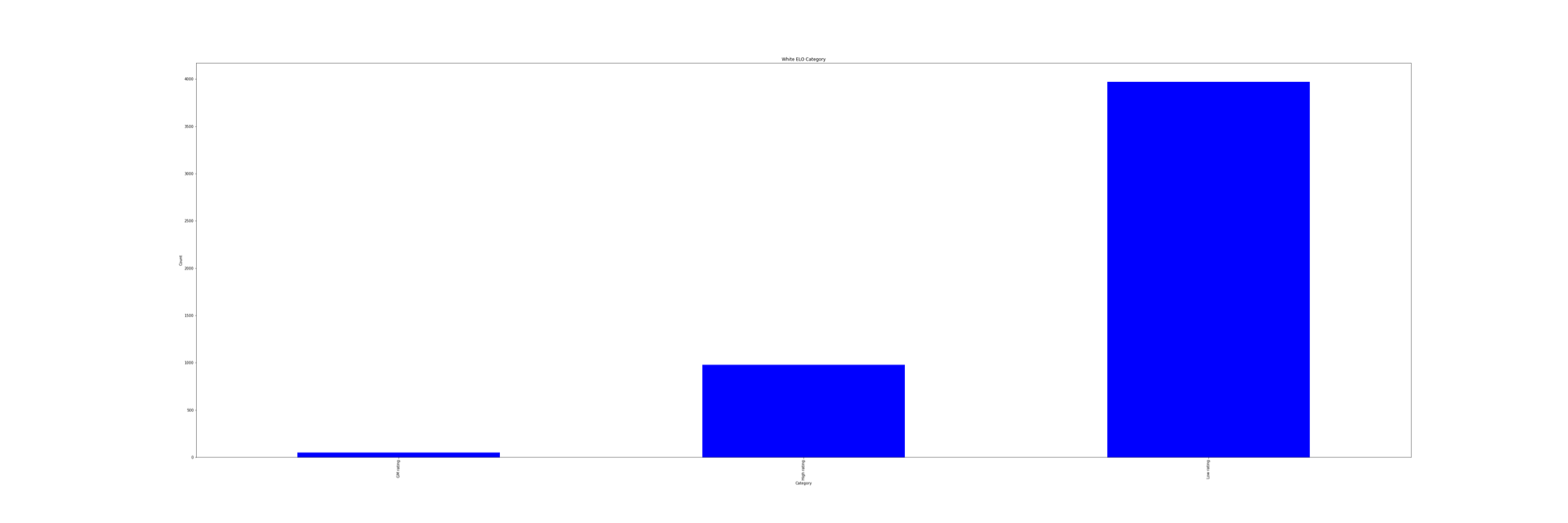
1. **Bar Charts**

From the bar charts conducted on the categorical variables, some descriptive details of the dataset were brought to light. To start with **Figure 8**, it can be seen that the most common game type in the dataset was Blitz. Additionally, from **Figure 9 and 10**, we can see that both the White\_elo\_category and the Black\_elo\_category most often fell in the low rating category. Finally, from **Figure 11**, we can find that the most common termination was classified as normal with time forfeit being the only other classification with any events.

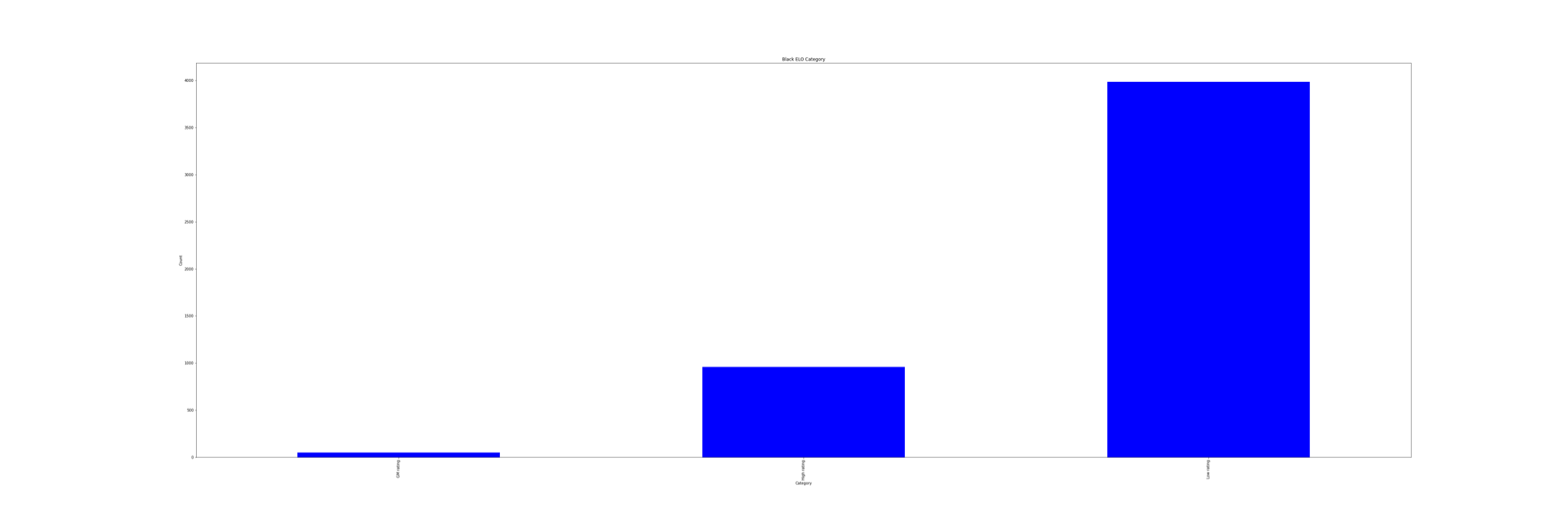
**Figure 8**

****

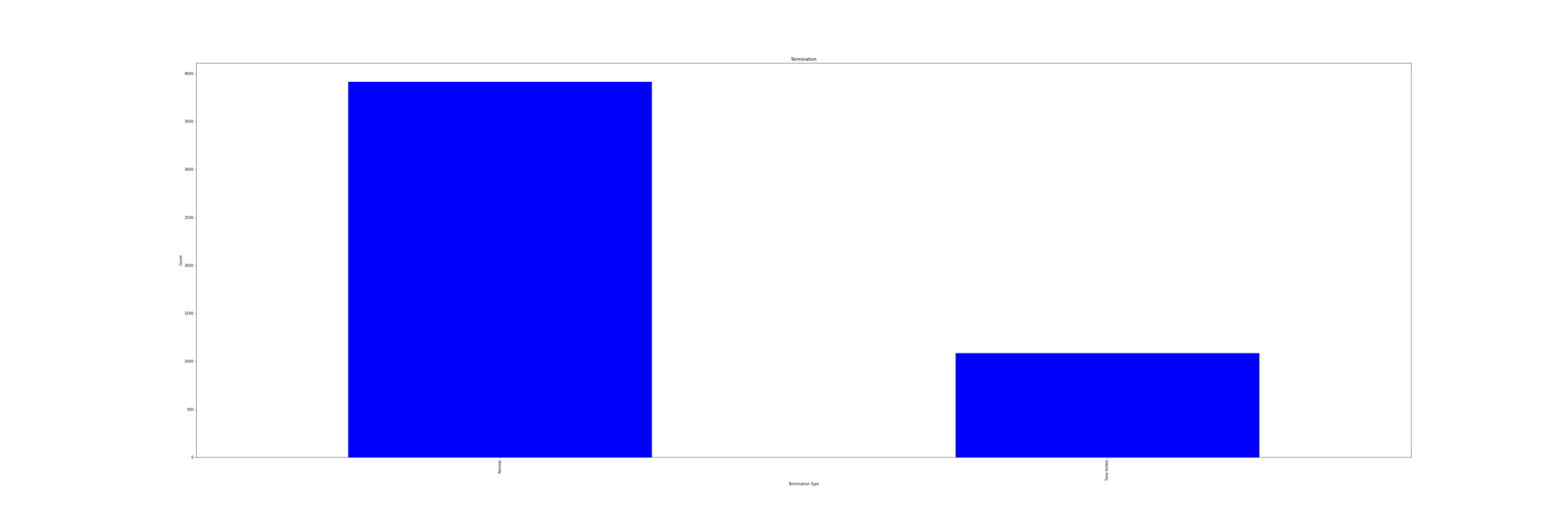
**Figure 9**

****

**Figure 10**

****

**Figure 11**

****

**METHODS**

1. **Data Preparation**

Prior to importing our data into Jupyter Notebook, we added two columns and removed millions of games from the dataset, leaving the dataset with only 4,988 games played on September 1, 2020. The two columns added were ‘White\_Result’ and ‘Black\_Result.’ These columns were added through Excel’s LEFT and RIGHT functions, as the initial Result column was imported into Excel as either the date 1/1/2020 or the number 36526. In adding these columns, we changed the results to either equal 0 for loss, 1 for win, and 0.5 for draw. Once imported into Python, we dropped any instances in which the result of the game was a draw, leaving us with only win or loss. Since this column was imported as a float64 data type, we converted it to an int64 datatype for the purposes of running the K-Nearest Neighbor model. We also removed the columns ‘Result’, ‘Opening’, ‘Site’, and ‘Event’ as they were irrelevant to the purpose of the project and were to specific to each game instance to keep in. Finally, we added a ‘Rating Difference’ column to remove any negatives and NaN results in the ‘WhiteElo’ and ‘BlackElo’ columns.

1. **Experimental Design**

**Table 3: Experiment Parameters**

|  |  |
| --- | --- |
| **Experiment Number** | **Parameters** |
| 1 | All four (4) raw / normalized features with 80/10/10 split for train, validate, and test. |
| 2 | All four (4) raw / normalized features with 70/15/15 split for train, validate, and test. |
| 3 | All four (4) raw / normalized features with 60/20/20 split for train, validate, and test. |

1. **Tools Used**

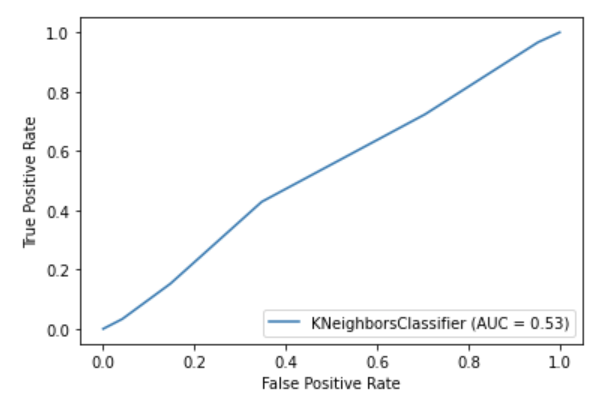
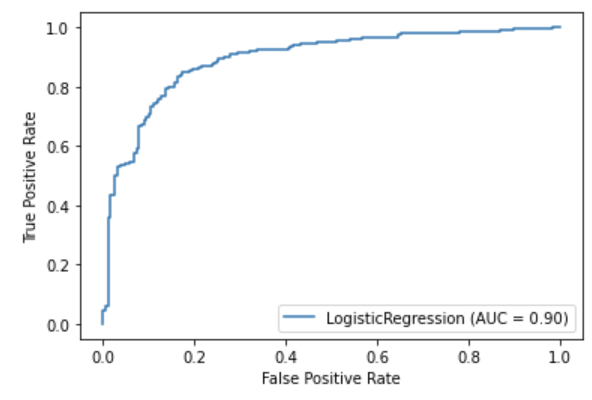
The following tools were used for this analysis: Python v3.5.2 running the Anaconda 4.3.22 environment for Apple Macintosh and Microsoft Windows 10 computer was used for all analysis and implementation. In addition to base Python, the following libraries were also used: Pandas 0.18.1, Numpy 1.11.3, Matplotlib 1.5.3, Seaborn 0.7.1, SKLearn 0.18.1. Pandas was used to create the Dataframe and run the exploratory analysis. Numpy was used to create the arrays that stored the data for the KNN test. Matplotlib was used to create the graphs for the graphical exploration in the data analysis. SKLearn was used to run the machine learning analysis for K-Nearest Neighbor, as well as the Logistic Regression and the Cross Validation.

**RESULTS**

1. **Classification Measures**
2. **Experiment 1:**

For the first experiment we ran, we used the 80/10/10 method which selected the data by choosing 80% for the trained values, 10% for the validated values, and 10% for the test values. The prediction model had an accuracy of 0.529, a precision of 0.489, a recall of 0.419, and an f1 of 0.451. The test model had accuracy of 0.535, a precision of 0.577, a recall of 0.429, and an f1 of 0.492. The Logistic Regression had an accuracy of 0.529, a precision of 0.479, a recall of 0.419, and an f1 of 0.451. The cross validation gave scores ranging between 0.796 AND 0.866. The logistic regression can be seen in **Figure 12** and the ROC curve can be seen in **Figure 13.**

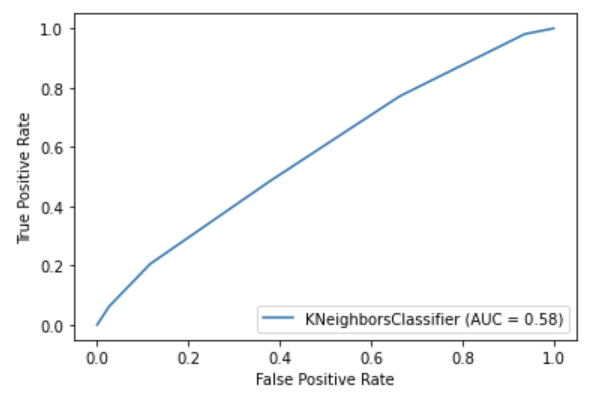
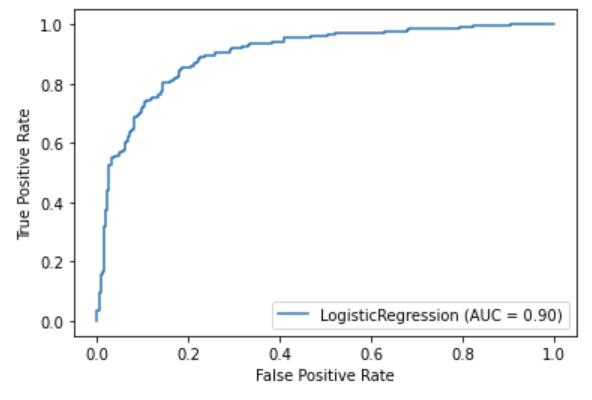
**Figure 12** **Figure 13**



1. **Experiment 2:**

For the first experiment we ran, we used the 70/15/15 method which selected the data by choosing 70% for the trained values, 15% for the validated values, and 15% for the test values. The prediction model had an accuracy of 0.521, a precision of 0.484, a recall of 0.425, and an f1 of 0.453. The test model had accuracy of 0.554, a precision of 0.553, a recall of 0.484, and an f1 of 0.517. The Logistic Regression had an accuracy of 0.521, a precision of 0.484, a recall of 0.425, and an f1 of 0.453. The cross validation gave scores ranging between 0.805 and 0.869. The logistic regression can be seen in **Figure 14** and the ROC curve can be seen in **Figure 15**.

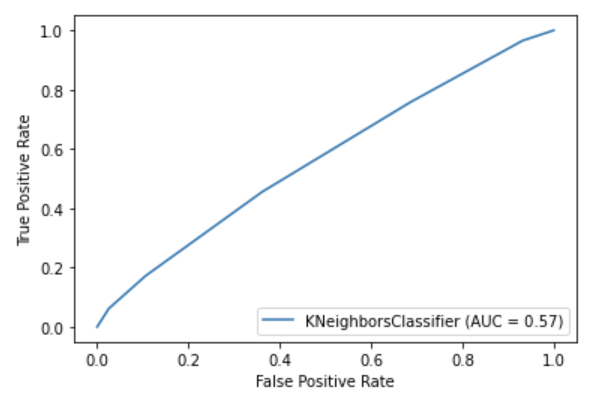
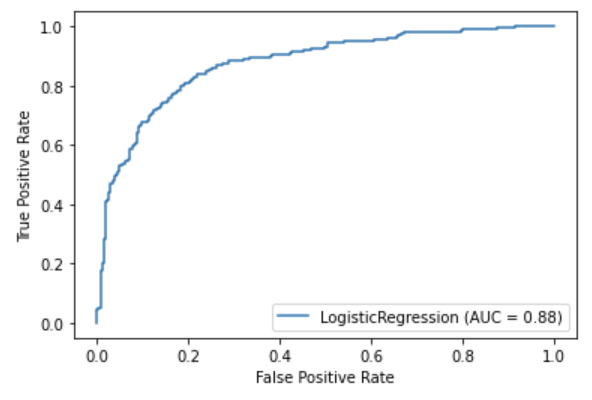
**Figure 14** **Figure 15**



1. **Experiment 3:**

For the first experiment we ran, we used the 60/20/20 method which selected the data by choosing 60% for the trained values, 20% for the validated values, and 20% for the test values. The prediction model had an accuracy of 0.531, a precision of 0.511, a recall of 0.438, and an f1 of 0.472. The test model had accuracy of 0.550, a precision of 0.541, a recall of 0.455, and an f1 of 0.494. The Logistic Regression had an accuracy of 0.531, a precision of 0.511, a recall of 0.438, and an f1 of 0.472. The cross validation gave scores ranging between 0.808 AND 0.896. The logistic regression can be seen in **Figure 16** and the ROC curve can be seen in **Figure 17.**

**Figure 16** **Figure 17**



1. **Discussion of Results**

For all of the experiments performed, there were no accuracies higher than 0.531, which indicates from the start that these models are not the best for the dataset used. Additionally, the precision scores never exceed 0.511 for any of the models, once again indicating that the models do not fit the dataset. These low precision levels indicate that there could be higher false positive rates. None of the recalls were higher than 0.5, which indicates that there is a low ratio of correctly predicted positive observations to all observations. Finally, the highest F1 value was 0.472, which is also a low value for scoring the false negatives and positives. Based solely on the accuracy, precision, recall, and F1 values, the best model in these experiments would be the 60/20/20 model as it has the highest value for all of those parameters. The worst model, if there can be a “worst model” when they are all generally bad, is the 70/15/15 model as it has the lowest precision and accuracy values, as well as rather low recall and F1 values.

1. **Problems Encountered**

The main issue with this project was finding a way to deal with the draws. After hours of toiling with different formats, lambda expressions, it was best to drop the rows entirely leaving a win/loss dichotomy. Another issue was the idea of openings, whether our analysis should have relied on pure move statistics such as blunders, or if we should have split the massive dataset into multiple data-frames, say one for each specific opening, and then performed our analysis.

1. **Limitations of Implementation**

The number one issue was dealing with the date instances from excel. When initially imported, we messed with the results column, splitting it into 0, .5, or 1. It looked like a string from there, so we spent a lot of time trying various different conversion methods from str to float, none of which panned out because it was a date the whole time. This was the main limitation. To get around this, we had to drop all of the draws, and as a result, we lost a fair amount of data. This isn’t as bad as it seems though, because while chess is infamous for its draws, the only players who consistently draw are the highest echelon, which is a fraction of the player base. Additionally, due to the number of chess games being played on any given day is so abundant, the selection we could make while still maintaining a functioning computer ended up all occurring on the exact same day which could have skewed the data as well.

1. **Improvements/Future Work**

It is upsetting that we could not find a workaround for the draws and openings. I think a more in depth analysis of those two aspects would be interesting. Many chess sites have percentile analysis on what openings win the most, which positions win the most, and in combination with a focus on move accuracy, this could have really proven to be an interesting study. The thing is, with chess, there’s a saying, “It’s never too late to be mated.” and this study is proof of that. A single blunder can be so disastrous that it can turn a game from a dominant, winning position to an instant mate in 1 move.

**CONCLUSION**

Overall, the K-nearest neighbor model for this chess dataset was overwhelmingly not the best model that could have been used for this specific set. That is not to say that K-nearest neighbor would not work for this dataset had the selection been more randomized or the data included been any more relevant. Despite not having the highest outputs for the machine learning analysis, we were able to find some interesting aspects about the games being played on the Lichess database, namely the most popular types of games and the ways that different moves can impact the overall outcome of a model. With the output we had from the K-nearest neighbor model being so low in accuracy values, as well as all of the other output values, it is clear that there needs to be more work done in order to form a more descriptive and accurate model for the dataset.

**REFERENCES**

K, George. “Lichess September 2020 Data.” *Kaggle*, 2 Mar. 2021, www.kaggle.com/noobiedatascientist/lichess-september-2020-data?select=Sept\_20\_analysis.csv.