Applied Analytics and Predictive Modeling: MGMT 6160
Darren Lin, Emily Pirie
5/8/2018

Go: What Makes a Winner?

Problem Definition:

An ancient game, with origins in China, the game of Go is one of strategy. In fact, Wikipedia states, 'Compared to chess, Go has both a larger board with more scope for play and longer games, and, on average, many more alternatives to consider per move'[1].

In Go, there are two opponents who take turns placing their own colored stones, either black or white, on the intersections of a 19x19 grid. The goal of the game is to capture your opponent's stones by placing your colored stones on either end of a string of the opponent's stones. Once stones are captured they are no longer in play. Play continues back and forth with each player either placing a black or white stone or passing on their turn. The game only ends when neither player wishes to make another move or by resignation of one of the players. Although there are several ways to score a round of Go, a common scoring method is the total of the number of empty intersections surrounded by one color plus all captured stones. As you can see below the area marked 'B' is a surround area by the black stones and thus these empty intersections will be counted as part of the black player's final score. Similarly the area labeled 'W' will all count towards the white player's score.

Image source: https://i.ytimg.com/vi/g-dKXOlsf98/maxresdefault.jpg

In this report we will look to create features from a dataset of over 500,000 recorded Go games played by professionals and amateur players in order to be able to answer a number of strategy related questions. Our hypothesis is that players that tend to win more often, have developed strategies to win. We will look to identify some of these strategies through our own hypothesized strategies and through looking for patterns in the data. The beginning strategies that will be utilized as a baseline are listed below as questions to how the given strategy might affect game play.

- Do winners have lower average move distances?
- Do winners start in a certain quadrant of the board?
- What features, if any, can help predict the winner before the game is over?

Data Sources & Feature Creation:

We obtained 535,721 saved Go games from the website: Joe's Go Database (https://pjreddie.com/projects/jgdb/) [2]. Each game is saved in a Smart Game Format (SGF) file, which means that we have access to each turn of the game and some limited metadata. Most of the files contain the outcome of the game and the location where each stone is placed for each turn of the game.

The file size for each file is between 0.6 to 2 kilobytes. The total size of all the games are approximately 2.2 gigabytes. We managed to compile the data into a single 90 megabytes CSV file using Python.

The format of the CSV file is such that each row represents a single game, and each column represents a feature that we felt were important to the analysis of the data. Some of the features were basic enough to simply extract from the SGF files, such as the rankings of the players and the outcomes of the games. The rest were more complicated and in general involved stepping through each turn of the game for more computations.

In order to be able to answer the questions we've identified around Go player strategies we have developed several features from the original dataset. This section will walk through each of created features and discuss the logic for each feature and how it might be used in modeling.

Location Conversion and Distance Calculation

Legal stone positions are recorded with 2 lowercase letters from 'a' to 's' where the first represents the column and the second represents the row on a standard 19x19 Go board. We converted the letters to numbers with respect to each letter's chronological order in the alphabet. In order to calculate distances, we used the distance formula ($\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$) derived from the Pythagorean Theorem using the stone locations as coordinates.

```
def coord_distance(a, b):
  return math.sqrt((ord(a[0]) - ord(b[0]))**2 +(ord(a[1]) - ord(b[1]))**2)
```

Move Count

We stepped through the game and kept a separate counter for black and white moves. For simplicity, we did not count handicapped moves towards the counters. Any games with fewer than 10 moves were considered early forfeits. Because we cannot fully explain why this might occur we have decided to remove these games from our dataset.

Average Subsequent Distance

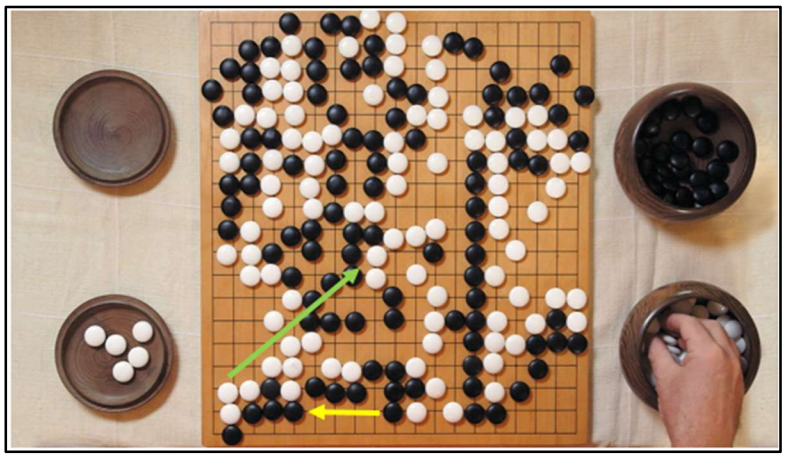
For each side, we summed the distances between each stone's location and the previous stone's location. Then we divided the summation by the total number of moves count of that side. We calculated both same side subsequent distances and opposite side subsequent distances.

```
black_dist_total/black_moves
white_dist_total/white_moves
```

```
black_dist_from_white_total/(black_moves-1) if black_moves > 1 else
  None, #black goes first
white_dist_from_black_total/white_moves if white_moves > 0 else None,
```

Due to the nature of placing different colored stones on a board in successive turns, we have hypothesized that the average distance between each of one player's subsequent moves will be highly correlated with a winning outcome. The idea being that winning players will intentionally stay close to their own colors (smaller average distance), focusing on one area of the board instead of placing pieces all over the board (larger average distance). A second related hypothesis is that each player will likely respond to the other player's move and will likely have a small distance in the piece that they play with relation to the other player's last piece.

To test this hypothesis we have created four related variables. The average distance over all pieces placed for each player. Specifically an average distance for black and an average distance for white. The second set of variables are the average distance for black from the previous white player's move and an average distance for white form the precious black player's move.



Illustrated above is an example of exactly how the distance is measured for each player. For the black opponent the distance from the first stone placed and the subsequent stone is 5 spaces. As discussed earlier, the distance was measured using the formula ($\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$).

Board Quadrants

We divided the board into four quadrants and assigned each corner of the Go board to a quadrant.

```
def coord_quadrant(a):
    x = ord(a[0])-96
    y = ord(a[1])-96
    # upper left: 1
    if x <= 10 and y <= 10:
        return 1
    # upper right: 2
    elif x > 10 and y <= 10:
        return 2
    # lower left: 3
    elif x <= 10 and y > 10:
        return 3
    # lower right: 4
    elif x > 10 and y > 10:
        return 4
```

We subtracted 96 from the x and y values because the letter ‘a’ converts to 97 according to the standard ASCII table. Since the board is 19x19, we wanted to offset row and column ‘a’ to 1. We subtracted 96 here and not in the distance formula because the offsets do not matter when both points have the same offsets.

For this feature the board is split into four quadrants as shown below. Please note that the quadrants are not completely even. Because the board is 19x19 the cutoffs were so that the first 10 columns are to the left and the first 10 rows are the top quadrants. This therefore leads to a slightly larger quadrant for #1 and a smaller quadrant for #4.

1	2
3	4

The hypothesis we developed around the starting quadrants is that winner's will likely stay within one or two quadrants over the first three moves of the game. To measure this, six variables were generated from the historic game plan data. The six variables are black_t1_quad, black_t2_quad, black_t3_quad, white_t1_quad, white_t2_quad and white_t3_quad. Manipulation of these variables will be necessary to determine if the order of the quadrants played in is significant to winning the game. As a quick note, we looked at including more quadrants, however this made for too many variables that it would be hard to find patterns.

Unique quadrants

For each side, we binned the turn counts by 10 and calculated the number of unique quadrants in which that side has had at least a stone placed over those 10 turns. In the CSV file, the data columns are formatted as <*color*>*_unique_quads_*<*starting move start*><*ending move count*>. For example, white_unique_quads_181190 means the number of unique quadrants that white has played a stone between turns 181 and 190, inclusive.

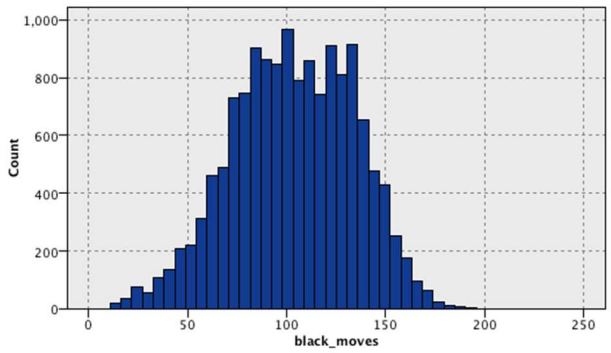
Past Performance Rating

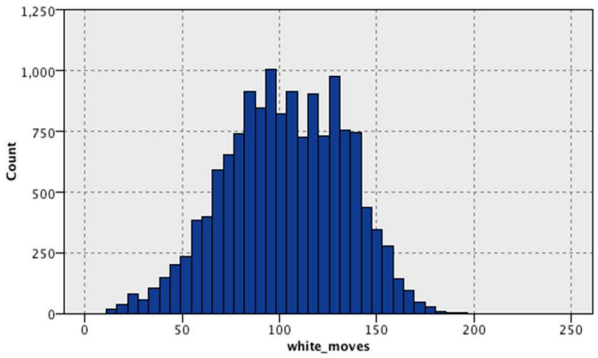
The performance rating for each player in each recorded game was an easy feature to create but a hard feature to use. Due to the nature of the recordings some of the data for this field has clearly been hand recorded. Although there are many levels to rankings in Go[4], as will be discussed later, the data came in with 614 unique rankings for the black players and 557 unique rankings for the white players. Some of the examples of what the data looks like is below. As you can see, many of the fields have names and ranking positions included when we were expecting to have just the rankings. A big data cleanup effort is still ahead for this feature.

[755] 25th Honinbo	9p, Kisei
[757] 1k*	6p*
[759] Judan	4p, Female Honinbo
[761] 9p, honorary Tengen	1p & 9p
[763] 8p, Oza	Honorary Gosei
[765] 24th Honinbo	5a
[767] Ama.	7p*
[769] 9p, Meijin, Honinbo	NHK Cup
[771] Daiwa Cup Grand Champion	8p, Kansai Ki-in 1st
[773] Gisung	1d prov.
[775] 9p, Judan	7p, Fujitsu Cup
[777] Judan	8p, Gosei
[779] Female Kisei	5a
[781] 9p, Judan	25th Honinbo
[783] Oza	9p, 25th Honinbo
[785] 8p & 7p	9p, Kisei
[787] Honinbo	9p, Honorary Kisei
[789] 4p & Judan	9p, Agon Cup
[791] 9p, 25th Honinbo	4p & Tengen, Judan
[793] 6p & 9p	5p, Female Honinbo
[795] Female Kakusei	Female 2crown
[797] Female Myeongin	Wangwi
[799] 9d & 4d	9p, 25th Honinbo
[801] 9p, Gosei	Haifong Cup

Data Analysis:

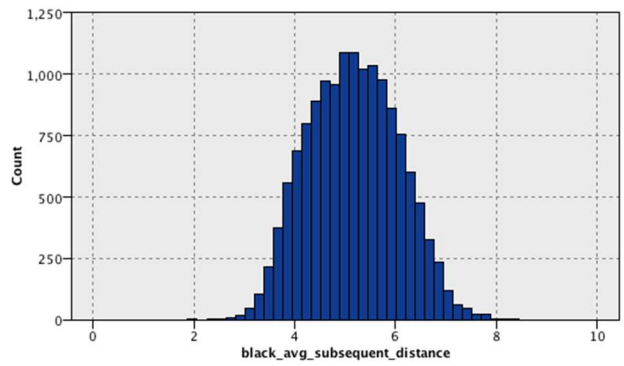
We conducted our data analyses using SPSS Modeler. Because we generated the CSV files ourselves, there were very few missing values (less than 3%), so we opted to ignore them. Overall, we found that very few games had over 200 moves from a side, even though the theoretical minimum of the longest game can last at least 10^{48} moves [3]. The average number of moves per game for both black and white in our data are mostly normally distributed, centered around 100 moves.

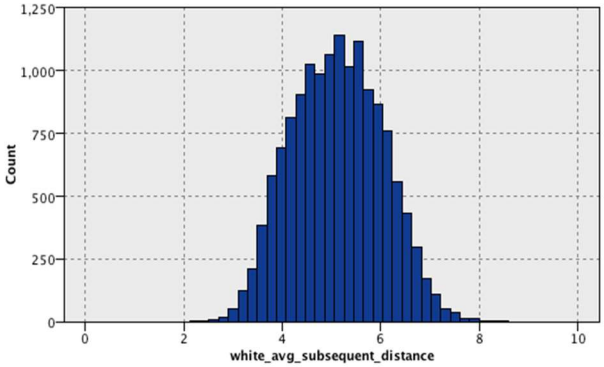




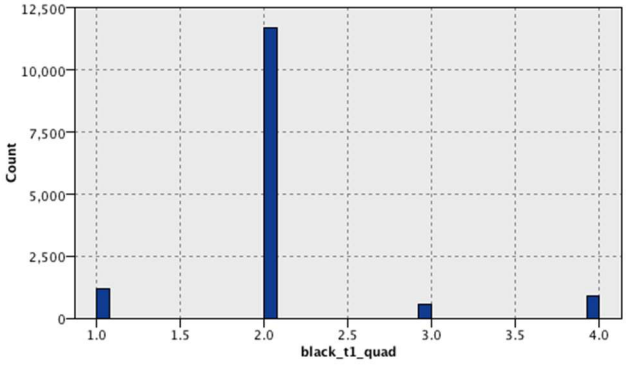
During our initial data exploration process we found that there seems to be quite a large number of games where there are suspiciously few moves. The game files do not contain additional information about the games, so we guess that these are games where one side decided to forfeit very early in the game. We removed those games with 10 or fewer moves from either side.

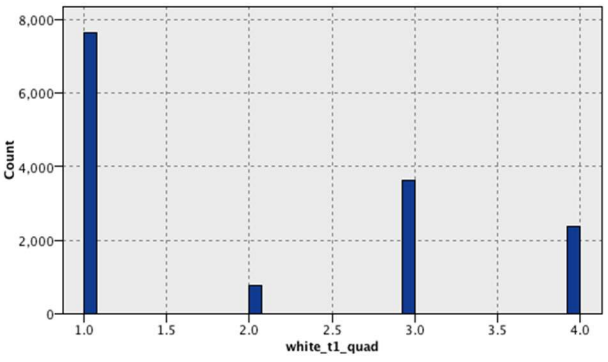
We also looked at same side subsequent distances for both sides and found no clear differences in distribution; both sides seem to have the same shape and are normally distributed.

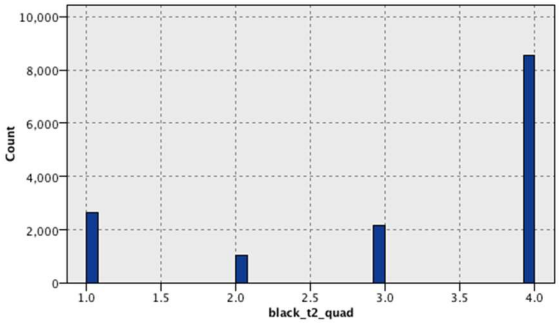


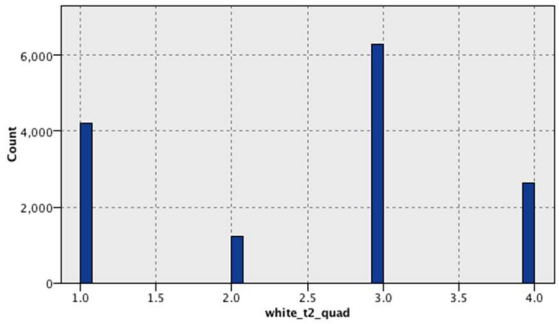


One of the goals of this project is to explore patterns in how players open their games. We found that players usually like to establish a “side” of the board as their territory. The following graphs show the distributions of the quadrants of the first three stones placed from each side.









In Go, black goes first, and we see that the majority of players start in quadrant 2 (top left image, top right of board), and the majority of players immediately respond by placing a stone in quadrant 1 (top right image, top left of board). Fewer players respond by placing a stone in quadrant 3 (bottom left of board, diagonal to quadrant 2). Similarly, in turn 2, black usually claims quadrant 4 and white usually claims the leftover quadrant 3. We can see that most of the time quadrant 2 and quadrant 4 (right side of board) are claimed by black and quadrant 1 and 3 are claimed by white (left side of board).

Finally, the last bit of interesting pattern that we found involved unique quadrants utilized as games progressed. We noticed that initially stone placements are very spread out. At the start of most games (turns 1-10), both black and white are placed in at least 3 different quadrants. We found earlier that players usually expand to have presence in at least two quadrants in the first two turns. We think that the reasoning behind such high count of unique quadrants in the first ten turns is because players want to gauge their opponent's reactions by placing stones in foreign territory. We see this aggressive pattern immediately subside in the next ten to twenty turns because we see players increasing focus on mostly two quadrants during this time. At around turn 31-40 we begin to see players spread focus across more than two quadrants again. The pattern of playing into more than three quadrants usually continues until the end of the game.

Modeling Approach/Method:

Based on the data exploration and created features there are three models we have decided to use for predicting future winners. The outcome variable: winner, either 'B', black, or 'W', white, wins the game or thus the models chosen are suited for binary outcomes. The models which will be discussed in greater detail later in this section are a linear and logistic regressions, weighted k-nearest neighbor and a decision tree.

Preprocessing the Data:

1) Before beginning the modeling process there were some data transformations that were needed to facilitate more accurate modeling. The outcome variable, winner, is transformed to a 1 for 'B' winners and a 0 for 'W' winners. This will be how we view the outcome variable for the remainder of the report.
2) There are 614 unique ratings for the black players and 557 unique rating levels for the white players in just a sample of the training data. To address the complication of having so many levels we developed a binning structure based of traditional guidelines for rankings [5]. Source: https://en.wikipedia.org/wiki/Go_ranks_and_ratings

Range	Stage	Bin
30 - 20k	Beginner	5

19-10k	Casual player	4
9-1k	Intermediate amateur	3
1-7d	Advanced amateur	2
1-9p	Professional player	1

This was actually quite a complicated step as there were so many individual entries for the ranking for each black and white player. A sample of the code for updating just the black player rankings into bins is below:

```
sample$group.b.ranking <- ifelse(sample$black_rating == '30k'| sample$black_rating == '29k'| sample$black_rating == '28k'
                              | sample$black_rating == '27k'| sample$black_rating == '26k'| sample$black_rating == '25k'
                              | sample$black_rating == '24k'| sample$black_rating == '23k'| sample$black_rating == '22k'
                              | sample$black_rating == '21k'| sample$black_rating == '20k', "5",
                             ifelse(sample$black_rating == '9k'|sample$black_rating == '10k' |sample$black_rating == '1k',
                                    '4',
                                    ifelse(sample$black_rating == ''| sample$black_rating == '1k'
                                           | sample$black_rating == '2k'| sample$black_rating == '3k'
                                           | sample$black_rating == '4k'| sample$black_rating == '5k'
                                           | sample$black_rating == '6k'| sample$black_rating == '7k'
                                           | sample$black_rating == '8k'| sample$black_rating == '9k', '3',
                                           ifelse(sample$black_rating == '5d'| sample$black_rating == '6d' |
                                                    sample$black_rating == '7d ama prov.'|sample$black_rating == 'L3d'
                                                  | sample$black_rating == '1d' | sample$black_rating == '2d'
                                                  | sample$black_rating == '3d'| sample$black_rating == '4d'
                                                  | sample$black_rating == '7d'| sample$black_rating == '8d'
                                                  | sample$black_rating == '9d'| sample$black_rating == '10d'
                                                  | sample$black_rating == '6a'| sample$black_rating == '3d ama'
                                                  | sample$black_rating == '4d ama'| sample$black_rating == '1d ama'
                                                  | sample$black_rating == '2d ama'| sample$black_rating == '5d ama'
                                                  | sample$black_rating == '6d ama'| sample$black_rating == '7d ama'
                                                  | sample$black_rating == '4d ama'| sample$black_rating == 'ama'
                                                  | sample$black_rating == '1a' | sample$black_rating == '2a'
                                                  | sample$black_rating == '3a'| sample$black_rating == '4a'
                                                  | sample$black_rating == '7a'| sample$black_rating == '8a'
                                                  | sample$black_rating == '9a'| sample$black_rating == '10a'
                                                  | sample$black_rating == '8d & 9d'| sample$black_rating == '6d, 3d, 2d'
                                                  | sample$black_rating == '2k ama'| sample$black_rating == '7d ama, 5p'
                                                 ,'2',
                                                 ifelse(sample$black_rating == '9' | sample$black_rating == 'Female Meijin & 9p'
                                                        | sample$black_rating == '7p, HA Cup'| sample$black_rating == '6p & Honorary Tengen'
                                                        | sample$black_rating == '6p, Female Kisei'| sample$black_rating == '9p,Tengen'
                                                        | sample$black_rating == '9d, Tengen, Oza'| sample$black_rating == '1p'
                                                        | sample$black_rating == '2p'| sample$black_rating == '3p'
                                                        | sample$black_rating == '4p'| sample$black_rating == '5p'
                                                        | sample$black_rating == '6p'| sample$black_rating == '7p'
                                                        | sample$black_rating == '8p'| sample$black_rating == '9p'
                                                        | sample$black_rating == '9p, Honinbo'| sample$black_rating == 'Tengen'
                                                        | sample$black_rating == '9p, Meijin'| sample$black_rating == 'Meijin'
                                                        | sample$black_rating == '3p, Female Saikyo'| sample$black_rating == '9p, Tengen'
                                                        | sample$black_rating == '39p, NEC Cup'| sample$black_rating == '3p, Female Kisei'
                                                        | sample$black_rating == '9p, NEC Cup'| sample$black_rating == '5p & 9p'
                                                        | sample$black_rating == 'Kisei'| sample$black_rating == '3p, Female Honinbo'
                                                        ,'1',0)))))
```

3) Similarly to the rating variable, the variable that tracks the average subsequent distance from the opponent's last piece has over 184,000 possible values. We therefore began to bin these distances it ranges as follows:

Range	Bin
<1	0
>=1 & <2	1
>=2& <3	2
>=3 & <4	3
>=4 & <5	4

>=5 & <6	5
>6	6

Linear & Logistic Regression

The linear and logistic regressions are arguably the most interpretable of models and is thus why we began with these models. Below are the outputs of each of our models. The top row showcase the linear regression results and on the bottom row displays the logistic regression outputs.

Because the process of binning each of the separate rating classifications was so manual we wanted to get a feel for how much of an impact it was to have both the black and the white ratings versus just one over the other. As it turns out the impact is not so great. The first iteration of the linear model shows statistical significance of a level of 0.05 or less for the level of Advanced Amateur, Intermediate Amateur and Casual Player. Additionally all other control variables, black and white move count, black and white average subsequent turn distance as well as black and white first turn quadrant are deemed statistically significant. Between the two iterations the only real difference is that the rankings for the black players change in significance. None of the white player rankings appear to be significant when the black player rankings are also included.

The logistic regression makes more sense for our specific scenario as it looks to provide coefficients that tell us the probability of the observation being an outcome of 1. Or in our case of the outcome being that the black player won. A similar situation appears to be true for the logistic regression that was true for the linear model so far as variable importance. In both models we see a slight improvement in the accuracy measures of Adjusted R squared and AIC.

Iteration #1: Only Black Player Rankings Binned

```
Call:
lm(formula = sample$winner ~ group.b.ranking + black_moves +
    white_moves + black_avg_subsequent_distance + white_avg_subsequent_dist
    black_t1_quad + white_t1_quad, data = sample)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2405 -0.3173  0.1163  0.2359  1.6713

Coefficients: (1 not defined because of singularities)
                              Estimate Std. Error  t value Pr(>|t|)
(Intercept)                  0.3468935  0.0577832    6.003 1.94e-09 ***
group.b.ranking0             0.0979874  0.0581522    1.685 0.091987 .
group.b.ranking1             0.0906888  0.0575036    1.577 0.114774
group.b.ranking2             0.1392731  0.0574913    2.423 0.015415 *
group.b.ranking3             0.1133172  0.0576377    1.966 0.049297 *
group.b.ranking4             0.2056854  0.0624306    3.295 0.000986 ***
group.b.ranking5                   NA         NA       NA       NA
black_moves                  0.4748037  0.0013434  353.446  < 2e-16 ***
white_moves                 -0.4741267  0.0013411 -353.532  < 2e-16 ***
black_avg_subsequent_distance  0.0054637  0.0021134    2.585 0.009731 **
white_avg_subsequent_distance -0.0470641  0.0020957  -22.457  < 2e-16 ***
black_t1_quad                -0.0144161  0.0013464  -10.707  < 2e-16 ***
white_t1_quad                 0.0127039  0.0006934   18.321  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4064 on 249651 degrees of freedom
Multiple R-squared:  0.3383,    Adjusted R-squared:  0.3382
F-statistic: 1.16e+04 on 11 and 249651 DF,  p-value: < 2.2e-16
```

Iteration #2: Black and White Rankings Binned

```
Call:
lm(formula = sample$winner ~ group.b.ranking + group.w.ranking +
    black_moves + white_moves + black_avg_subsequent_distance +
    white_avg_subsequent_distance + black_t1_quad + white_t1_quad,
    data = sample)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2405 -0.3175  0.1157  0.2360  1.6714

Coefficients: (2 not defined because of singularities)
                              Estimate Std. Error  t value Pr(>|t|)
(Intercept)                  0.1185761  0.4104468    0.289 0.772662
group.b.ranking0             0.1437210  0.0583271    2.464 0.013738 *
group.b.ranking1             0.1444374  0.0579780    2.491 0.012730 *
group.b.ranking2             0.1340997  0.0574858    2.333 0.019662 *
group.b.ranking3             0.1106025  0.0577496    1.915 0.055467 .
group.b.ranking4             0.2116804  0.0625658    3.383 0.000716 ***
group.b.ranking5                   NA         NA       NA       NA
group.w.ranking0             0.1310514  0.4064873    0.322 0.747150
group.w.ranking1             0.1727869  0.4064309    0.425 0.670741
group.w.ranking2             0.2329693  0.4063446    0.573 0.566422
group.w.ranking3             0.2308307  0.4064005    0.568 0.570043
group.w.ranking4             0.0783470  0.4130464    0.190 0.849559
group.w.ranking5                   NA         NA       NA       NA
black_moves                  0.4748149  0.0013473  352.433  < 2e-16 ***
white_moves                 -0.4741455  0.0013451 -352.508  < 2e-16 ***
black_avg_subsequent_distance  0.0056301  0.0021143    2.663 0.007748 **
white_avg_subsequent_distance -0.0467386  0.0020969  -22.290  < 2e-16 ***
black_t1_quad                -0.0144180  0.0013463  -10.710  < 2e-16 ***
white_t1_quad                 0.0126197  0.0006937   18.192  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4063 on 249356 degrees of freedom
Multiple R-squared:  0.3386,    Adjusted R-squared:  0.3385
F-statistic:  7977 on 16 and 249356 DF,  p-value: < 2.2e-16
```

```
Call:
glm(formula = sample$winner ~ group.b.ranking + black_moves +
    white_moves + black_avg_subsequent_distance + white_avg_subsequent_dis
    black_t1_quad + white_t1_quad, family = "binomial", data = sample)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8279  -0.7818  -0.2517   0.6660   3.5479

Coefficients: (1 not defined because of singularities)
                              Estimate Std. Error  z value Pr(>|z|)
(Intercept)                  -1.980149   0.509743   -3.885 0.000102 ***
group.b.ranking0              1.505155   0.511149    2.945 0.003233 **
group.b.ranking1              1.513184   0.508426    2.976 0.002918 **
group.b.ranking2              1.735409   0.508401    3.413 0.000641 ***
group.b.ranking3              1.530695   0.509085    3.007 0.002640 **
group.b.ranking4              1.983502   0.533318    3.719 0.000200 ***
group.b.ranking5                    NA         NA       NA       NA
black_moves                   2.547799   0.009885  257.757  < 2e-16 ***
white_moves                  -2.544183   0.009867 -257.843  < 2e-16 ***
black_avg_subsequent_distance 0.043441   0.013078    3.322 0.000895 ***
white_avg_subsequent_distance -0.237713   0.012973  -18.323  < 2e-16 ***
black_t1_quad                -0.084314   0.009149   -9.216  < 2e-16 ***
white_t1_quad                 0.058544   0.004284   13.667  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 345665  on 249662  degrees of freedom
Residual deviance: 248045  on 249651  degrees of freedom
AIC: 248069

Number of Fisher Scoring iterations: 4
```

```
Call:
glm(formula = sample$winner ~ group.b.ranking + group.w.ranking +
    black_moves + white_moves + black_avg_subsequent_distance +
    white_avg_subsequent_distance + black_t1_quad + white_t1_quad,
    family = "binomial", data = sample)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8279  -0.7827  -0.2523   0.6655   3.5469

Coefficients: (1 not defined because of singularities)
                              Estimate Std. Error  z value Pr(>|z|)
(Intercept)                  1.928e+11  6.161e+11    0.313 0.754278
group.b.ranking0             1.814e+00  5.481e-01    3.309 0.000937 ***
group.b.ranking1             1.856e+00  5.458e-01    3.401 0.000672 ***
group.b.ranking2             1.776e+00  5.410e-01    3.282 0.001030 **
group.b.ranking3             1.530e+00  5.433e-01    2.816 0.004869 **
group.b.ranking4             2.061e+00  5.468e-01    3.770 0.000163 ***
group.b.ranking5                    NA         NA       NA       NA
group.w.ranking0            -1.928e+11  6.161e+11   -0.313 0.754278
group.w.ranking1            -1.928e+11  6.161e+11   -0.313 0.754278
group.w.ranking2            -1.928e+11  6.161e+11   -0.313 0.754278
group.w.ranking3            -1.928e+11  6.161e+11   -0.313 0.754278
group.w.ranking4            -1.928e+11  6.161e+11   -0.313 0.754278
group.w.ranking5            -1.928e+11  6.161e+11   -0.313 0.754278
black_moves                  2.547e+00  9.904e-03  257.169  < 2e-16 ***
white_moves                 -2.543e+00  9.887e-03 -257.249  < 2e-16 ***
black_avg_subsequent_distance  4.435e-02  1.309e-02    3.389 0.000703 ***
white_avg_subsequent_distance -2.367e-01  1.298e-02  -18.231  < 2e-16 ***
black_t1_quad                -8.415e-02  9.144e-03   -9.203  < 2e-16 ***
white_t1_quad                 5.808e-02  4.287e-03   13.550  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 345268  on 249372  degrees of freedom
Residual deviance: 247683  on 249355  degrees of freedom
AIC: 247719

Number of Fisher Scoring iterations: 25
```

Weighted k-Nearest Neighbor

The next model we evaluated for the Go dataset is a weighted k-means nearest neighbor model. This model looks to take data in a high dimensional space and look towards the other observations closest to the point and determine the outcome of the specific observation based on the ones around it. By training this model we are able to validate the most optimal number of neighbors to look at over the entire dataset.

```
Call:
train.kknn(formula = winner ~ group.b.ranking + group.w.ranking +     black_avg_subsequent_distance
 + white_avg_subsequent_distance,     data = sample, kmax = 7)

Type of response variable: continuous
minimal mean absolute error: 0.3598024
Minimal mean squared error: 0.2590859
Best kernel: optimal
Best k: 7
```

We find that the trained model indicates that using the 7 closest neighbors will yield the best results. A mean squared error of 25.9% is predicted based on this trained kknn model. We can now take this trained model and run it against our validation dataset to see how well the kknn, using the 7 nearest neighbors, predicts the winner of the game.

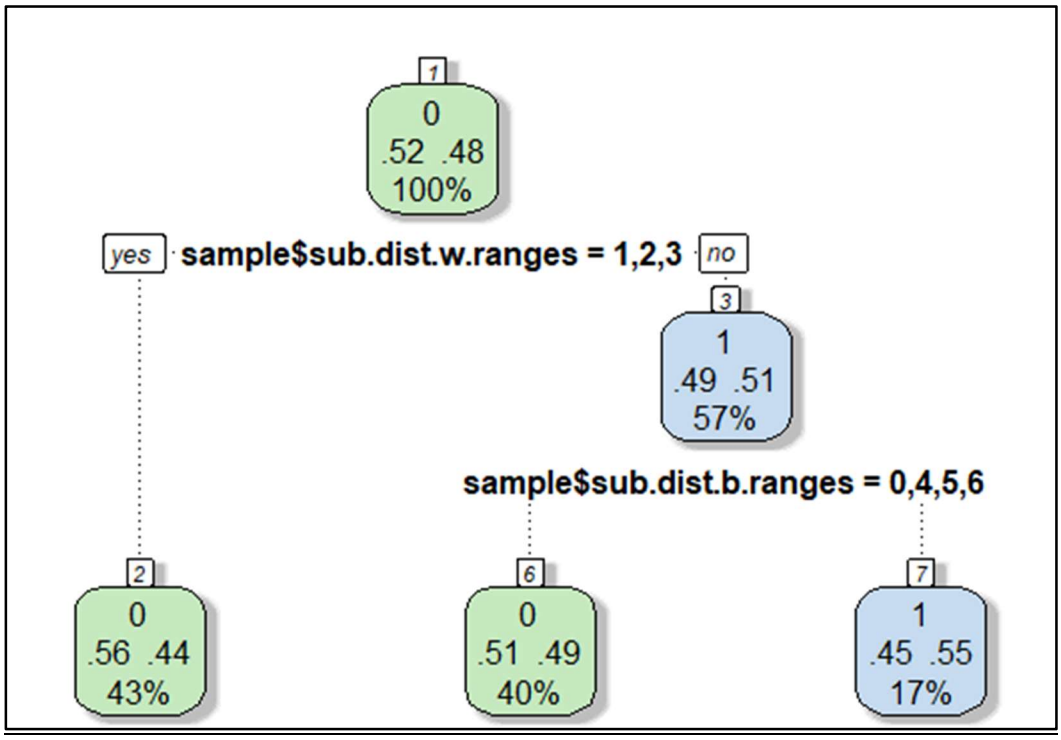
Overall we find a model with accuracy of 53.8%. The following is a table showing the predicted outcome vs the actual outcome as recorded in the dataset.

Weighted k-Nearest Neighbor Confusion Matrix:

Winner	Prediction	
	0 - White	**1 - Black**
0 - White	14760	115142
1 - Black	0	119471

Decision Tree

The decision tree has many benefits, variable selection/reduction and interpretability. For these reasons we decided to run a decision tree model on the Go dataset. Based on this model the overall accuracy achieved is 53.9%. What we learn from this tree is that in general, the second player tends to have a lower distance between their placed stone and their opponents just completed stone placement. When there is a smaller distance between these pieces for the second player, they tend to win more often.



Decision Tree Confusion Matrix:

Winner	Prediction	
	0 - White	**1 - Black**
0 - White	111289	18587
1 - Black	96400	23061

Conclusion/Findings:

Overall our findings are that the strongest indicators of success are the number of moves and the overall ranking of the player coming into the game. These intuitively make sense. A higher ranked player has already shown success in the past and is thus likely to be successful in the future. A quick note that we might be facing some autocorrelation issues with this variable. Additionally it makes sense that if a player can get even just one extra move in that their probability of a win should be improved. The logistic regression model indicated that for just one more move, the probability of black winning increased by 2.54%.

Additionally we found that the majority of players start in quadrant 2 and the majority of players immediately respond by placing a stone in quadrant 1. Similarly, in turn 2, black usually claims quadrant 4 and white claims the leftover quadrant 3. We can see that most of the time quadrant 2 and quadrant 4 (right side of board) are claimed by black, the first player, and quadrant 1 and 3 are claimed by white (left side of board).

Lastly we found that when the second player achieves a lower overall distance from their opponent's previous move, they win more often than not. We hypothesize that this is because the second player (white stones) is more often able to play offensive in this position.

Reference:

1| Go (game). (2018, May 04). Retrieved May 8, 2018, from https://en.wikipedia.org/wiki/Go_(game)
2| Redmon, J. (n.d.). Joe's Go Database. Retrieved May 8, 2018, from https://pjreddie.com/projects/jgdb/
3| Number of Possible Go Games. (n.d.). Retrieved May 8, 2018, from https://senseis.xmp.net/?NumberOfPossibleGoGames
4| EGF Rating System. (n.d.). Retrieved May 8, 2018, from https://senseis.xmp.net/?GoR
5| Go ranks and ratings. (2018, April 10). Retrieved May 8, 2018, from https://en.wikipedia.org/wiki/Go_ranks_and_ratings

Appendix:

All associated files are stored on GitHub here: https://github.com/darrendlin/mgmt6140-final