



Predicting March Madness

Jordan Hyatt | Avery Hutchinson | Ross Prevatt | Richard Sangster

Introduction

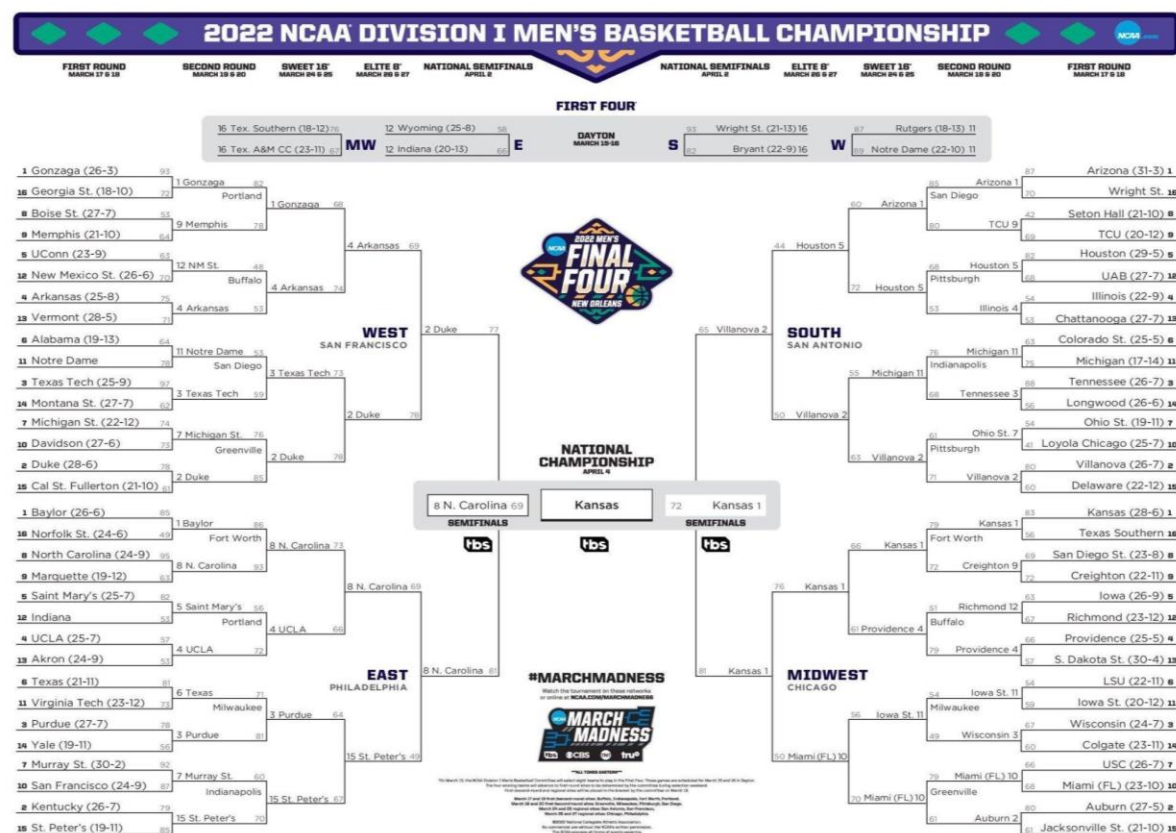
Sports gambling is a massive industry that spans around the globe. It includes basically any sport that is played with American Football, Horse Racing, Soccer, Baseball and Basketball being amongst the most popular to bet on. The global sports betting industry as a whole reached a market size of 231 billion US dollars in 2022. This, of course, only includes legally gambled money and does not include illicit bets being placed worldwide.

Given the sheer amount of money being made/lost in sports gambling there is a compulsive desire amongst all gamblers to gain a competitive advantage in predicting the outcomes of contests. Unfortunately, sometimes this manifests in cheating. Using insider knowledge to gain an advantage or worse altering/influencing the outcome of the contest also known as “fixing”. Another legal route, which will be the focus of this project, is to gain a competitive advantage through data analytics.

With the advancement of computing and networking technology the capacity to keep, store, distribute and analyze sports data and statistics continues to grow. This has been occurring for many decades now. There are extremely large and robust data repositories available to the public for all kinds of sports. With this much data available it seems like an obvious idea to attempt to use it to gain an advantage in predicting the outcome of contests. This is something that is being done by many people world wide. This project sets out to do exactly that for the NCAA March Madness Tournament.

“March Madness” is the nickname given to the annual NCAA Men’s basketball tournament who’s winner is crowned National Champions. The tournament involves 64 teams and is a single elimination format. To no one's surprise the tournament takes place in March. The event is nationally televised and is the subject of many office pools where people fill out a “bracket”. Filling out a bracket is essentially attempting to predict the outcome of each game all the way down to the National Champion. To date there has never been an official bracket filled out perfectly. This is not surprising considering if you picked teams at random your chances of perfect bracket are 1 in 9.22 quintillion. A picture of a bracket from the 2022 tournament is shown below (figure 1).

Figure 1:



The intent of this project is to utilize publicly available data and apply modern machine learning and data mining techniques to create a model that can predict the outcome of games more accurately than not just random picks but also of an “informed” gambler or sports analyst.

Analyses

Data Source

The data for this project came from Kaggle.com's march madness competition. The data in its raw form as well as more details on the competition can be found at this url.


<https://www.kaggle.com/competitions/mens-march-mania-2022/data>.

Data Content

The data set as a whole is quite large and complex. It comes as a collection of csv files, many of which provide supporting information for the main dataset which provides the statistics of each game of every NCAA season going back to 2003. A list of all the csv files used in this project and their descriptions are given in the table below.

file	desc
cities.csv	listing of city/state where games were played in and their corresponding id
conferences.csv	listing of ncaa conferences and their corresponding abbreviation
Mseasons.csv	A listing of all the seasons and the date it started (day 0)
Mteams.csv	A listing of all the NCAA mens basketball teams and their respective ids
Mcoaches.csv	A listing of all the NCAA mens basketball coaches per team per season
MTeamConferences.csv	A listing of the conference each team belonged to per season
MNCAATourneySeeds.csv	Seeding data for every NCAA MM tournament
MMasseyOrdinals_thruDay128.csv	Listing of every teams ranking per ranking system per day per season
MGameCities.csv	The city each game was played in
MRegularSeasonDetailedResults.csv	Detailed stats for every regular season game
MNCAATourneyDetailedResults.csv	Detailed stats for every tournament game

The dataset in its final form has 68 dimensions and includes 1,466 rows of data. The entire data set incorporates statistics of previous games played in the March Madness tournament, going back to 2003. The different statistics recorded include team ids, game



ids, rebounds, steals, blocks, win/loss, home/away, assists, etc. Due to the robust nature of this data set, it is perfect to use many different models and methods to attempt to discover any unseen patterns, in order to accurately predict a bracket for the upcoming 2021 March Madness tournament which is the most recently completed tournament for which we have data.

Data Cleaning

Since the data was sourced from kaggle it is extremely well organized and clean. Very little is required as far as cleaning the raw data. However, due to the sheer size, complexity and format of the data there is much to be done in terms of modeling, processing and transforming the data into a dataset than can be used in machine learning algorithms.

Data Model

Because of the size, complexity and format of the data it would be extremely difficult to process, combine, transform and aggregate the data using traditional scripting methods. Therefore, our team decided to first model the data in a way that allows for the generation of a relational database schema. We can then process, combine, transform and aggregate the raw data as it is stored in a database. From there, it is easy to recall the data and build up a dataset that is appropriate for applying machine learning techniques to. The Entity Relationship Diagram (ERD) describing the data model and schema is shown in the figure below.


```

class City(models.Model):
    ...''' Represents a city a game could be played in '''
    ...kid = models.IntegerField(verbose_name="Kaggle's Unique ID", unique=True)
    ...name = models.CharField(max_length=200)
    ...state = models.CharField(max_length=2)

    ...@classmethod
    ...def update_objs(cls):
    ...    '''A method to update cities from csv'''
    ...    path = os.path.join(settings.BASE_DATA_PATH, 'Cities.csv')
    ...    df = pd.read_csv(path)
    ...    for tup in tqdm(df.itertuples()):
    ...        obj, _ = cls.objects.update_or_create(
    ...            kid=tup.CityID,
    ...            defaults=dict(
    ...                name=tup.City, state=tup.State
    ...            )
    ...        )

    ...def __str__(self):
    ...    return f'{self.name} | {self.state}'

```

The main table that represents the records/instances that machine learning can be applied to is GameTeam. Each instance of this model represents a game that was played (or will be played by a given team). For each instance there are ForeignKeys to game stats (for games that were already played). The stats are typical basketball metrics such as rebounds, shots, 3-pt FGs etc.

Unfortunately, for future games, these stats will not exist until after the game is played and will be useless as inputs to a predictive model. To rectify this feature engineering is required and described in the next section.

Feature Engineering

As described in the previous section, feature engineering is required in order to produce metrics that can be applied to games in the future that have yet to be played (the desired entity to be predicted on). The scheme used to engineer features was to average the statistics for a given team and their opponent for all games played that season up to the game being predicted. These produce inputs that can be described in future games. This was achieved through a series of model methods and scripts that were run on the data. The engineered features were stored in a NoSql style field (HStoreField on Postgres) called "features" on the GameTeam table.

Finalized Dataset

The final dataset consists of all tournament games played with the engineered features described in the previous section. It was filtered to only include tournament games since that is the desired games to be predicted on. It should be noted that although the dataset was filtered to only include tournament games, regular season games were still used in computing the engineered features. A small snippet of the finalized dataset is shown in the figure below.

id	game_id	team_id	opponent	loc	result	season	conf	avg_DR	avg_OR	avg_PF	avg_TO	avg_Ast	avg_BlK	avg_FGA	avg_FGM
206121	49184	320	143	N	0	2018	aec	24.5	9.53125	16.40625	11.6875	14.90625	2.46875	58.125	25.625
205317	16123	245	142	N	0	2012	big_ten	21.85294	11.14706	17.76471	8.705882	13.5	3.205882	57.91176	25.55882
205800	38076	118	43	N	1	2016	big_west	26.93333	10.43333	20.46667	13.16667	15.63333	3.9	56.46667	25.93333
206072	49158	53	109	N	1	2018	aac	25.85294	13.05882	15.82353	11.47059	15.97059	5.529412	58	26.26471
205203	10755	250	142	N	0	2011	a_ten	23.86111	9.111111	16.02778	10.33333	15.02778	4.25	53.94444	25
205785	38066	292	146	N	0	2016	aec	26.5	11.83333	16.23333	11.66667	16.3	4.6	58.93333	28.1
205550	27072	214	244	N	1	2014	acc	25.6875	14.0625	20.3125	12.03125	15.5625	4.78125	59.875	27.71875
205392	21552	293	185	N	1	2013	big_east	24.2	14.54286	16.17143	12.62857	14.54286	6.2	58.37143	25.68571
206419	64107	325	111	N	0	2021	pac_twelve	25.84375	10.21875	5.15625	12.03125	14.03125	5.21875	57.96875	27.5
205990	43646	214	16	N	1	2017	acc	27.67647	15.88235	17.76471	11.82353	18.20588	3.235294	66.17647	31.05882
205537	27065	318	66	N	0	2014	sun_belt	24.19355	13.64516	21.16129	13.87097	13.09677	3.935484	59.74194	27.3871
205150	10704	226	327	N	1	2011	big_ten	23.29412	11.20588	14.26471	10.17647	15.61765	3.323529	55.58824	27.44118
206207	54758	109	122	N	0	2019	sun_belt	24.59375	8.03125	17.46875	11.78125	12.125	4.6875	57.375	26.375
206015	43679	362	111	N	0	2017	big_east	25.32432	11.83784	18.62162	12.62162	14.97297	2.621622	56.37838	25.91892

Data Models and Methods

With a finalized dataset in place, machine learning methods can be explored on the dataset to find the model best suited for predicting the outcome of NCAA March Madness games. A list of models/techniques to be explored are listed below.

- Clustering
- Naive Bayes
- Decision Trees
- Support Vector Classifier

The general strategy will be to apply the above models to the dataset tweaking input parameters until an ideal solution is realized. Since the dataset is perfectly balanced (50% Wins / 50% Losses) the metric to be used to grade the models will be accuracy. The ultimate goal is to identify a well performing model with specified parameters to make predictions on future NCAA March Madness Tournament games.

Results

This section will lay out the results obtained from exploring each of the 4 models/techniques listed in the previous section.

Clustering

This data set has such a high dimensionality, therefore euclidean distance is likely to perform inefficiently and will not provide the information and results we are looking for; a better choice is to use cosine similarity.

The data itself is already fairly clean since it was taken directly from Kaggle. The data needed to be transformed and prepared slightly for clustering to work and provide relevant results. Clustering requires numerical data to work, therefore, character needs to either be changed to numerical or integer type data, or the data should be set aside and not used in the clustering method.

By using clustering methods and distance measures, we can determine the level of similarity between data points. If we did not already have data labels (team 1, team 2, etc.) clustering would provide information on suggested groups certain data points should belong to. Clustering allows the data to be grouped together based on likelihood of relativity to other data points. If done correctly, each team's data should be grouped together and easily discernible from other teams. Certain clustering techniques also allow for the grouping results to be easily visualized as an image.

With the original data set being so large, a good option is to cut the data into a smaller dataset by using only the tournament related data and cutting out the regular season data. While this does create a smaller data set, it is still too large to return easily understood images of the clusters. Therefore, another cut of the data can be performed to only include 5 teams. Since not every team makes it into the tournament, the count function can be used to determine which teams made it into the tournament, as well as how many observations there are for each team. In The figure below, team 24 has 24 rows of data and team 1 has only 3 rows of data meaning that team 24 played more tournament games than team 3 did. Using this method we can create a small data set of 5 teams with high "games played" numbers. An example would be teams 12, 24, 29, 81 and 111.

```

team_id  n
1      3
3      2
4      6
6      1
7      4
10     1
11     1
12    20
13     4
14     3
15     2
16     9
20     7
22     1
24    24
25     6

```

Now that the data set is more manageable, we can start to perform distance measures and clustering analysis to see any differences or similarities between the data points. The image below represents the first 7 rows of the data matrix compared to each column and the results of the cosine similarity distance matrix performed on the data set. The smaller the number, the less distant the two data points are from one another; a 0.00 result means that the data points are exactly the same. Most of the data points in the image below show similarity between the data points, but no exact matches. However, the image is only a snippet of the comparison results because the data set is so large.

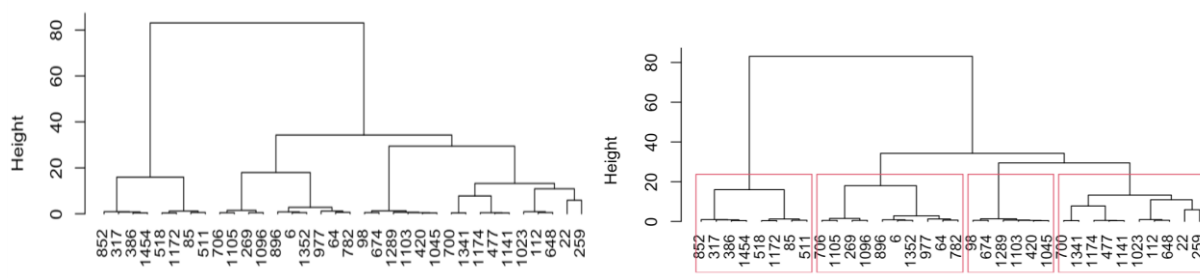
```

> print(distMatrix_C2)
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
[1,] 1.0000000 0.9989915 0.9988114 0.9986451 0.9988181 0.9988200 0.9968763 0.9999930 0.9968930 0.9979763
[2,] 0.9989915 1.0000000 0.9997323 0.9992582 0.9997182 0.9993309 0.9985777 0.9990262 0.9985911 0.9989100
[3,] 0.9988114 0.9997323 1.0000000 0.9989987 0.9999958 0.9991371 0.9988023 0.9988387 0.9988349 0.9993616
[4,] 0.9986451 0.9992582 0.9989987 1.0000000 0.9989527 0.9999797 0.9975563 0.9986346 0.9974534 0.9975601
[5,] 0.9988181 0.9997182 0.9999958 0.9989527 1.0000000 0.9990903 0.9988520 0.9988494 0.9988881 0.9994141
[6,] 0.9988200 0.9993309 0.9991371 0.9999797 0.9990903 1.0000000 0.9977367 0.9988002 0.9976411 0.9976816
[7,] 0.9968763 0.9985777 0.9988023 0.9975563 0.9988520 0.9977367 1.0000000 0.9969158 0.9999932 0.9983185
      [,11]      [,12]      [,13]      [,14]      [,15]      [,16]      [,17]      [,18]      [,19]     [,20]
[1,] 0.9987679 0.9987840 0.9999737 0.9988685 0.9972552 0.9999738 0.9971772 0.9973157 0.9969887 0.9967781
[2,] 0.9992918 0.9993062 0.9989983 0.9997034 0.9975515 0.9989370 0.9973961 0.9976702 0.9986371 0.9985565
[3,] 0.9990505 0.9990422 0.9988685 0.9999799 0.9980961 0.9988203 0.9979698 0.9982267 0.9988814 0.9987786
[4,] 0.9999928 0.9999925 0.9986692 0.9988748 0.9956679 0.9985817 0.9955299 0.9957785 0.9976425 0.9974939
[5,] 0.9990032 0.9989985 0.9988811 0.9999893 0.9981609 0.9988347 0.9980330 0.9982944 0.9989301 0.9988268
[6,] 0.9999951 0.9999851 0.9988341 0.9990298 0.9960691 0.9987523 0.9959498 0.9961621 0.9978191 0.9976753
[7,] 0.9976107 0.9975762 0.9969769 0.9989583 0.9984788 0.9968888 0.9983972 0.9985034 0.9999936 0.9999974

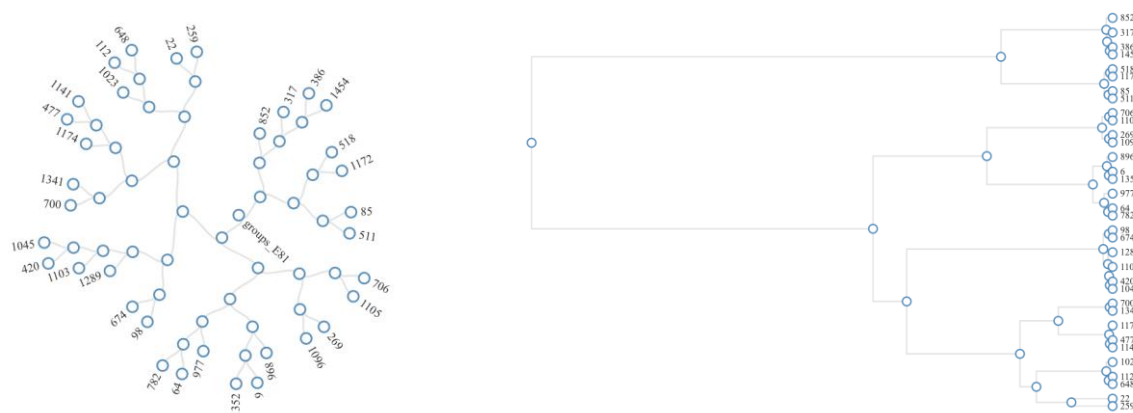
```

Distance measures can also be plotted to visually show differences and similarities between data points. In order for each point to show up clearly in the images below, the distance measures were run against team 81 only. This makes the data set even smaller, and it also can help highlight differences between games played by team 81. The numbers along the bottom of the images below represent the different rows of data which can also be known as the data for different games played. The below images are of hierarchical

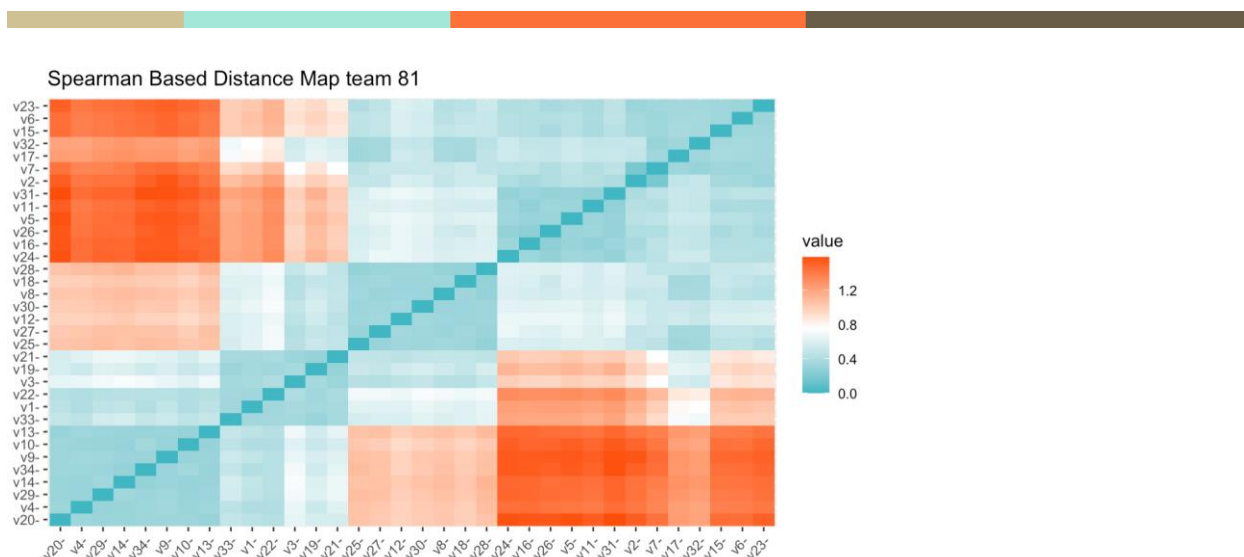
clusters which show where the distances are grouped together and the red boxes help in highlighting the similar clusters.



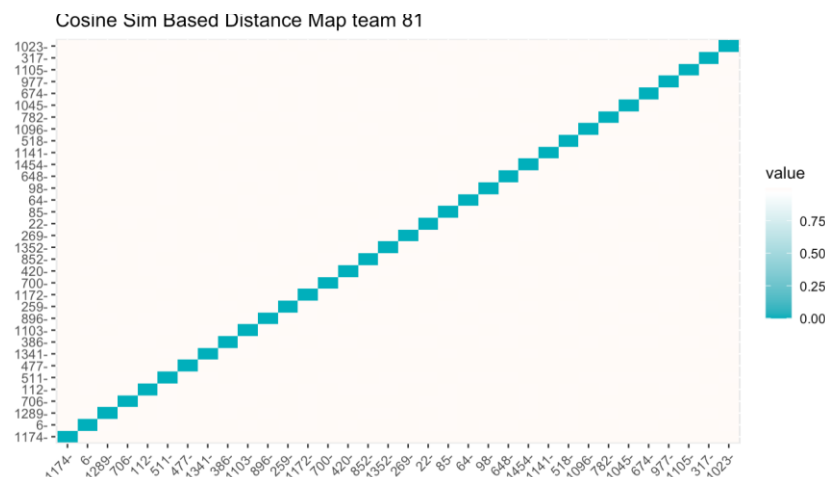
These results can also be shown in radial cluster form or as a traditional dendrogram like the images below.



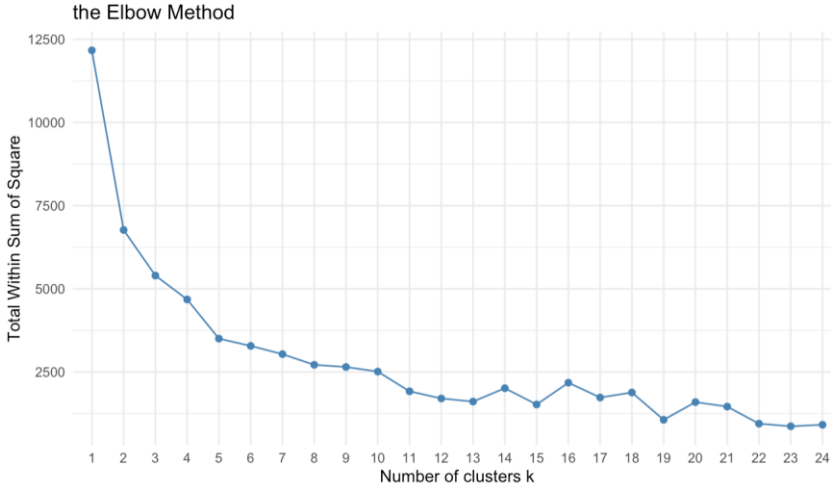
Applying a heat map to the distances is also a great way to visualize the distance data. Below, we can see a perfect diagonal line in dark blue. This means that the distance between the x axis data point and the y axis data point is 0 meaning that they are the same number and there is no difference. The orange data is for distance values higher than 1.2 and shows that the two data points are very different from each other and therefore a greater distance measure.



We can also use a cosine similarity heatmap to visualize distances in team 81 data. The cosine similarity heatmap does not show as great of difference in the scale of the distances but we can still see the perfect diagonal blue line that is representing the 0.00 distance line.



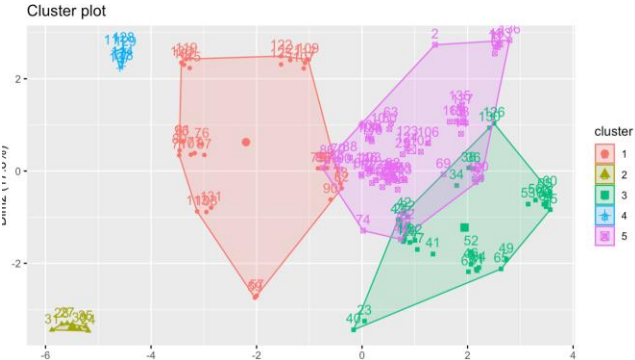
Another type of cluster is a k-means cluster, where k is a pre-defined number of clusters, and the data is sorted into the predefined number of groups by reducing the in-cluster sum of squares. It is possible to have a few large clusters, or even many small clusters. The hope and goal is to choose the correct number for k, cluster the data around the centroids (center of the cluster) and keep the centroids as small as possible. Below, we can see the “elbow method” algorithm and how it takes the given set and works to suggest the optimal number for k.



As we can see above, the optimal suggested number for k is 5. After 5 the slope of the line starts to run more horizontal than vertical. This is the point where k is optimal. Below are the results of the K-means algorithm using the suggested 5 clusters for k .

[illegible]

The below cluster plots show the difference between manually picking 4 centers or picking 5 centers.



```
> table(teams$team_id, k2results$cluster)
```

	1	2	3	4	5
12	17	0	0	3	0
24	12	0	0	6	6
39	5	2	0	16	0
81	20	14	0	0	0
111	17	13	6	0	0

in cluster 2. This is roughly a 60/40 split of the data meaning that the data points for team 81 did not all cluster together for some reason.

Some tuning can be done on the data if we wanted to get closer to 100 percent accuracy where each team's data was represented by only one cluster, but that tuning could also end up skewing the results and creating a sort of bias in the data. For the most part, clustering this data doesn't tell us too much about the data set because all of the numbers are already very similar. We can expect this result in this data set in particular because only the best teams with the highest averages end up going into the tournament anyways. And picking the 5 teams with the higher number of games played and therefore the higher number of observations, also means that the probability of their data being similar to one another is quite high. All in all, clustering tells us a few things about this data set, but it is not one of the better methods to choose for analysis.

Naive Bayes

The Naive Bayes model is a strong model for predicting complex problems using the assumption that the variables are independent of each other. The data within the march madness tournament could be dependent on other variables. There are high correlations between some of the variables which may affect how well the model can be created. Below are the correlations between Rank and Seed, Attempt and Made field goals, and Made field goals and Score.

RankvSeed	AttemptvMade	MadevScore
0.8293634	0.7713951	0.922418

Another factor of Naive Bayes is that its inputs like to be split into bins more than continuous structure. For this case the test was run with both to see if there is any difference between the two models. The discrete sections are split up using high, medium and low values. The splits are evenly distributed between the sections to cluster each of the values into a set.

The following continuous data is full of actual values or averages so the model sees the small discrepancies between values. This can also cause a problem as the Naive Bayes classifier works better with categorical data. The data may not run as well due to its own classification so the data has been broken down into categories.

avg_DR	avg_OR	avg_PF	avg_TO	avg_Ast	avg_Blkl	avg_FGA	avg_FGM	avg_FTA	avg_FTM	avg_Stl	avg_FGA3
28.60000	8.533333	4.533333	9.733333	17.13333	3.400000	63.00000	31.46667	19.60000	14.333333	7.200000	22.53333
22.45161	9.161290	6.709677	11.161290	14.09677	3.483871	55.74194	23.77419	19.32258	14.903226	5.612903	19.70968
22.04167	8.875000	8.083333	11.125000	11.33333	3.166667	57.29167	23.20833	18.91667	13.541667	7.541667	25.16667
24.00000	8.903226	6.322581	10.193548	13.41935	2.838710	57.93548	26.51613	18.22581	13.064516	4.967742	18.35484
25.96667	10.400000	4.933333	13.833333	14.13333	4.333333	64.56667	27.80000	18.46667	13.300000	8.633333	30.40000
21.86207	11.137931	7.172414	11.275862	16.86207	3.655172	62.72414	30.58621	16.62069	11.655172	8.931034	24.17241
27.00000	8.600000	8.350000	12.900000	14.15000	2.650000	54.85000	25.85000	19.00000	13.150000	4.000000	19.85000
21.44000	6.880000	4.320000	10.560000	16.00000	3.400000	58.64000	27.80000	17.00000	12.360000	9.160000	21.68000
23.26923	7.230769	6.000000	12.269231	13.07692	3.000000	53.50000	25.19231	14.50000	10.730769	6.730769	19.88462
25.55172	6.896552	4.103448	10.689655	15.93103	3.482759	59.75862	28.37931	15.75862	10.103448	6.965517	26.44828
28.27586	9.448276	7.000000	12.758621	16.31034	2.827586	58.86207	29.34483	23.06897	15.931034	5.724138	18.06897

Figure: Continuous data

avg_DR	avg_OR	avg_PF	avg_TO	avg_Ast	avg_Blkl	avg_FGA	avg_FGM	avg_FTA	avg_FTM	avg_Stl	avg_FGA3
high	medium	medium	low	high	medium	high	high	medium	medium	medium	medium
low	medium	high	medium	medium	medium	low	low	medium	high	low	low
low	medium	high	medium	low	medium	medium	low	medium	medium	medium	medium
medium	medium	medium	low	low	medium	medium	low	medium	medium	low	low
medium	medium	medium	high	medium	high	high	medium	medium	medium	high	high
low	medium	high	medium	high	medium	high	high	medium	low	high	medium
high	medium	high	medium	medium	low	low	low	medium	medium	low	low
low	low	low	low	medium	medium	medium	medium	medium	medium	high	medium
low	low	medium	medium	low	medium	low	low	low	low	medium	low
medium	low	low	medium	medium	medium	medium	medium	low	low	medium	high
high	medium	high	medium	medium	medium	medium	medium	high	high	low	low

Figure: Discrete data

Both Naive Bayes models used different training and testing data. The data was identical but the discrete training data was binned just like the testing data. The continuous data produced some solid results after running it against the testing data.

Below is a confusion matrix of the results. The continuous version did quite well. The accuracy was just as good as the discrete model and they produced very similar tables. The models performed just as well as each other but did not yield very strong results. The 65% accuracy does not help us out too much in our bracket as choices made just by assuming the higher seeded team would win, yielded a 67% accuracy.

```

      answers
nbPredict 0 1
0 43 23 AccuracyNB
1 23 43 0.6515152

```

Tables: Confusion Matrix and Accuracy for Continuous Results

```

      answers
nbPredict2 0 1
0 44 24 AccuracyNB2
1 22 42 0.6515152

```

Tables: Confusion Matrix and Accuracy for Discrete Results

```

      answers 0 1
0 42 24 AccuracySeed
1 19 47 0.6742424

```

Decision Tree

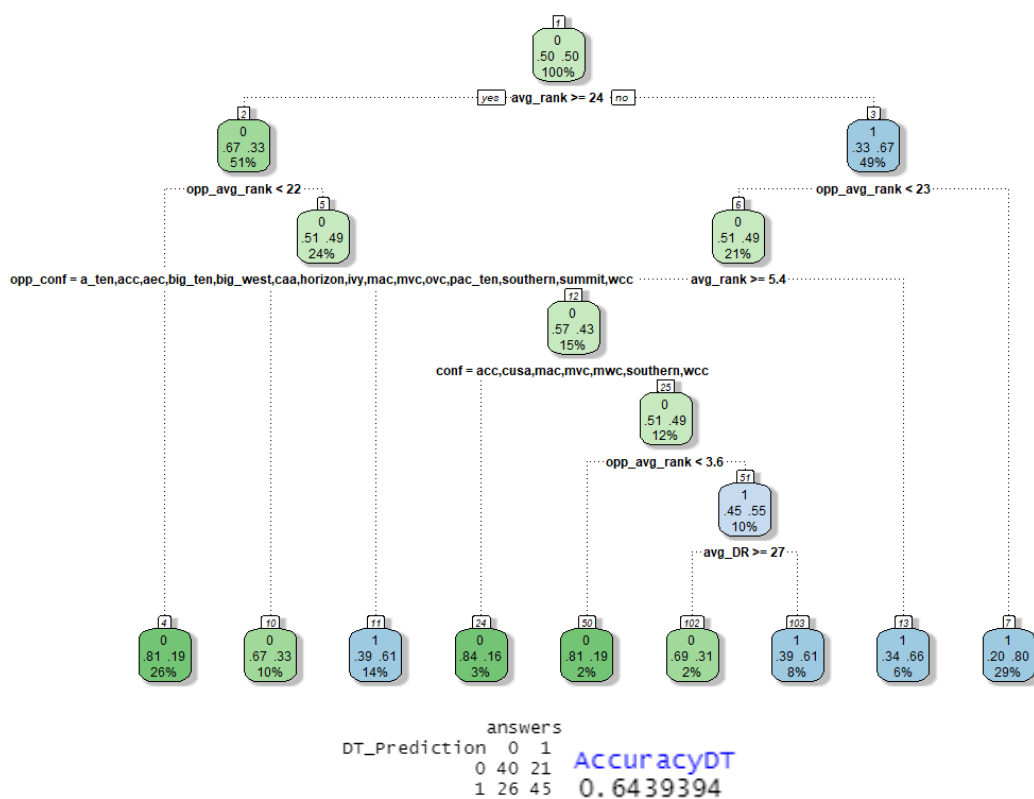
Decision Tree analysis is similar to the Naive Bayes model in the sense that it runs off a training model to predict a new set of data. The data is allowed to be continuous and will be kept that way for these next few models. The advantage of the decision tree is that it can visually show its decision making process. The decision tree shows the effects that a single decision has on the outcome and by looking at the variable importance, the most valuable attributes will be more prominent.

The decision trees did not perform as well as expected. This could be caused by the fact the data doesn't align very well due to its randomness in the games. The best performing of the three models was the version where only the seeds were utilized to predict with. Normally utilizing all the values would sound like the best option but this can also cause overfitting. The first decision tree is most likely overfitting the data but also with the randomness with the teams, it is very hard to predict who will win based on the stats.

Below are the results of the decision trees. The first block of data is the variable importance of the values used in the decision trees. The next is the decision tree and below that is the confusion matrix and accuracy.

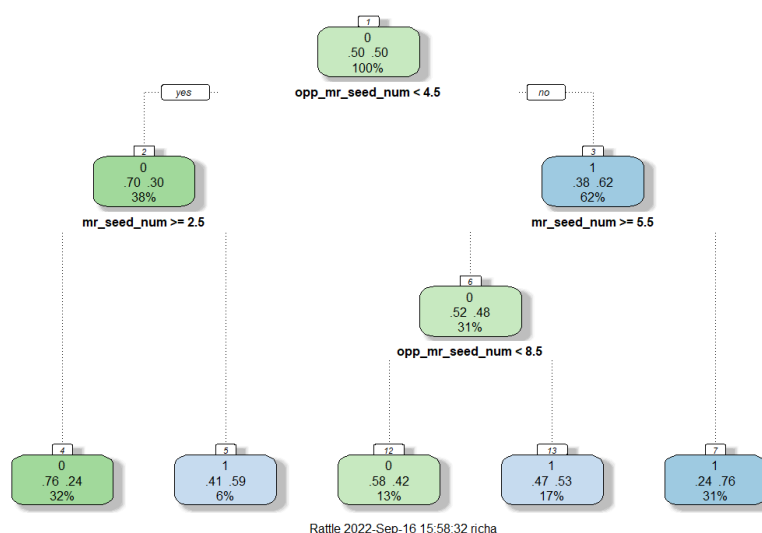
Decision Tree 1(all variables):

avg_rank	mr_seed_num	opp_avg_rank	conf	opp_mr_seed_num	opp_conf	avg_FGM
80.4861975	76.7301045	65.0785232	52.6797656	52.6332944	36.5398998	23.0768532
avg_Score	avg_DR_opp	opp_avg_DR_opp	opp_avg_Blk	opp_avg_FTM_opp	opp_avg_FGM	avg_DR
21.1796622	21.0462101	8.6030242	5.8319707	5.5753314	4.7337719	3.8584003
avg_FGA_opp	opp_avg_Score_opp	avg_Blk	avg_Stl	opp_avg_PF	opp_avg_FTA_opp	avg_Ast
1.6324001	1.4167615	1.3386556	1.2278600	1.2278600	1.0389585	0.8904001
opp_avg_FGA3_opp	avg_FGM_opp	avg_FGM3_opp	avg_FGA3_opp	opp_avg_FGM3	opp_avg_FGM3_opp	avg_Score_opp
0.8679877	0.7420001	0.7420001	0.6743448	0.6509908	0.6509908	0.5469031
avg_FGA3	avg_FTA	avg_OR_opp				
0.3646020	0.3646020	0.3646020				



Decision Tree 2 (both teams seeds):

`opp_mr_seed_num` 80.32770 `mr_seed_num` 63.78826



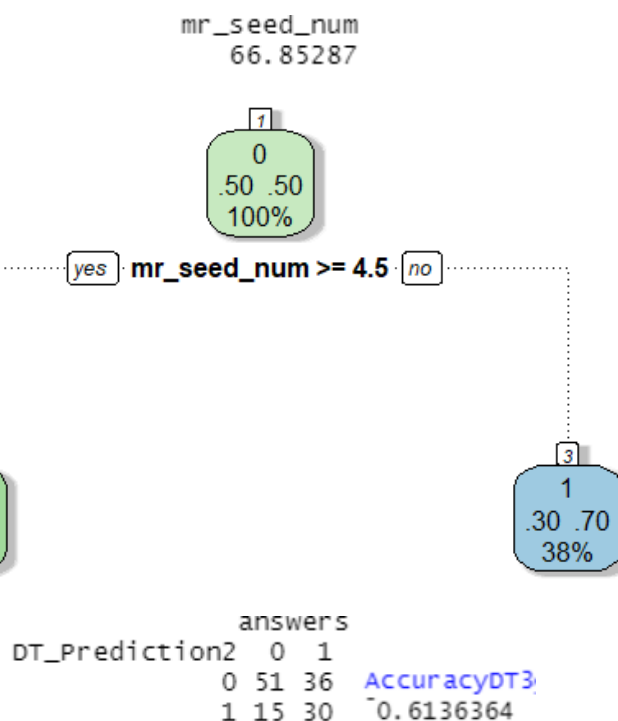
Rattle 2022-Sep-16 15:58:32 richa

```

answers
DT_Prediction2 0 1
                0 40 20
                1 26 46
AccuracyDT2
0.6515152

```

Decision Tree 3 (only rank)



Overall the decision trees did fairly well in prediction but they also did not beat the technique of just choosing the winner by the higher seed. The accuracy is much lower than expected.

SVM

SVM is another promising method to explore in terms of predicting the outcomes of games. This analysis was run using the SVM model from the “e1071” package in R, by varying kernels and cost measures to improve the results and efficiently. The kernels to be used are linear, polynomial, and radial. The objective here is to train the models on historical tournament game data with the goal of being able to predict the winners of the 2021 tournament games as accurately as possible.

A few extra data preparation steps were necessary for SVM analysis. Similar to previous models, the data set for building the model was limited to NCAA tournament games from 2010 to 2019, obtained from the original Kaggle data. The first step was to delete any unnecessary variables that may hinder the model’s results, such as the game ID, team ID, location, and season year. Variables related to conferences also had to be removed, as the conferences in the training data do not match the conferences in the 2021 tournament. Results and seed variables were then converted to factors for analysis. Finally, the data was split into training and testing data frames. The method used was to select every third

observation for inclusion in the test data set, with the remaining observations included in the training set. This creates a one-third, two-thirds split, with most of the observations used for training. Initially 889 observations were used to train the models, while 445 observations were used for testing.

9 initial models were trained and tested, using the 3 kernels previously mentioned along with 3 different cost measures of 0.2, 1, and 10. The table below summarizes the results by model run on the **test** data set. Each box represents a model (labeled at top) and contains a confusion matrix and the number of support vectors, as well as an accuracy calculation.

Initial Results:

Linear, Cost = 1						SupVect 562						Poly, Cost = 1						SupVect 487						Radial, Cost = 1						SupVect 889					
	L	W	Tot	Correct	290		L	W	T	Correct	275		L	W	T	Correct	259		L	W	T	Correct	259												
L	150	71	221	Incorrect	155		L	147	83	230	Incorrect	170		L	100	52	152	Incorrect	186		L	100	52	152	Incorrect	186									
W	84	140	224	Total	445		W	87	128	215	Total	445		W	134	159	293	Total	445		W	134	159	293	Total	445									
T	234	211	445	Accuracy	65%		T	234	211	445	Accuracy	62%		T	234	211	445	Accuracy	58%		T	234	211	445	Accuracy	58%									

Linear, Cost = 10						SupVect 545						Poly, Cost = 10						SupVect 487						Radial, Cost = 10						SupVect 889					
	L	W	T	Correct	288		L	W	T	Correct	275		L	W	T	Correct	273		L	W	T	Correct	273												
L	144	67	211	Incorrect	157		L	147	83	230	Incorrect	170		L	118	56	174	Incorrect	172		L	118	56	174	Incorrect	172									
W	90	144	234	Total	445		W	87	128	215	Total	445		W	116	155	271	Total	445		W	116	155	271	Total	445									
T	234	211	445	Accuracy	65%		T	234	211	445	Accuracy	62%		T	234	211	445	Accuracy	61%		T	234	211	445	Accuracy	61%									

Linear, Cost = 0.2						SupVect 575						Poly, Cost = 0.2						SupVect 487						Radial, Cost = 0.2						SupVect 889					
	L	W	T	Correct	302		L	W	T	Correct	275		L	W	T	Correct	211		L	W	T	Correct	211												
L	156	65	221	Incorrect	143		L	147	83	230	Incorrect	170		L	0	0	0	Incorrect	234		L	0	0	0	Incorrect	234									
W	78	146	224	Total	445		W	87	128	215	Total	445		W	234	211	445	Total	445		W	234	211	445	Total	445									
T	234	211	445	Accuracy	68%		T	234	211	445	Accuracy	62%		T	234	211	445	Accuracy	47%		T	234	211	445	Accuracy	47%									

The SVM models using a linear kernel proved to be the most accurate on the test data, with all 3 models outperforming those with polynomial and radial kernels. The most accurate model used a linear kernel with a cost of 0.2, coming in with 68% accuracy. Interestingly, that model also had the highest number of support vectors among the linear kernel models.

Next the models were re-trained using all the data from our initial dataset (combining the training and test data for a total of 1,334 observations). The models were then run on the 2021 tournament games that were saved to a separate data frame. This data contained 132 observations (66 games) with the same 61 variables. In essence the 2021 games became the new test data. Results are presented below.

Final Model Runs on 2021 Tournament Games:

Linear, Cost = 1						Poly, Cost = 1						Radial, Cost = 1							
SupVect 860						SupVect 710						SupVect 1334							
	L	W	T	Correct	85		L	W	T	Correct	69		L	W	T	Correct	80		
L	48	29	77	Incorrect	47		L	34	31	65	Incorrect	63		L	21	7	28	Incorrect	52
W	18	37	55	Total	132		W	32	35	67	Total	132		W	45	59	104	Total	132
T	66	66	132	Accuracy	64%		T	66	66	132	Accuracy	52%		T	66	66	132	Accuracy	61%

Linear, Cost = 10						Poly, Cost = 10						Radial, Cost = 10							
SupVect 710						SupVect 710						SupVect 1334							
	L	W	T	Correct	69		L	W	T	Correct	69		L	W	T	Correct	85		
L	34	31	65	Incorrect	63		L	34	31	65	Incorrect	63		L	29	10	39	Incorrect	47
W	32	35	67	Total	132		W	32	35	67	Total	132		W	37	56	93	Total	132
T	66	66	132	Accuracy	52%		T	66	66	132	Accuracy	52%		T	66	66	132	Accuracy	64%

Linear, Cost = 0.2						Poly, Cost = 0.2						Radial, Cost = 0.2							
SupVect 710						SupVect 710						SupVect 1334							
	L	W	T	Correct	69		L	W	T	Correct	69		L	W	T	Correct	66		
L	34	31	65	Incorrect	63		L	34	31	65	Incorrect	63		L	1	1	2	Incorrect	66
W	32	35	67	Total	132		W	32	35	67	Total	132		W	65	65	130	Total	132
T	66	66	132	Accuracy	52%		T	66	66	132	Accuracy	52%		T	66	66	132	Accuracy	50%

The new models were less accurate for predicting the 2021 tournament games. 2 models were able to achieve 64% accuracy - the linear kernel with a cost of 1 and the radial kernel with a cost of 10. Many models struggled with accuracy around 50%, obviously not adding value. For completeness, the best initial model for each kernel was also run on the 2021 games and they performed even worse. The most accurate initial model used a polynomial kernel with a cost of 1, but only achieved an accuracy rate of 51%. So it does appear the final models are more accurate.

Keep in mind that each game in our data set consists of two observations, one for each team playing. This allows for a greater sample size and keeps the data set balanced from a win/loss perspective. However, for the model to be run in practice, only one team (observation) should be used to avoid inconsistent results (model making the same prediction for each team). For example, the following chart shows the results of the best 2 models for 2021 tournament games with all observations included. Recall these models are 1) linear kernel with cost of 1 and 2) radial kernel with a cost of 10.

Detailed Results for Linear and Radial SVMs for 2021 Tournament Games:

	Kernel Used	
	Linear	Radial
Games Correctly Predicted	37	29
Games Incorrectly Predicted	18	10
Inconsistent Predictions	11	27
Total Games	66	66
% of Inconsistent Predictions	17%	41%
Accuracy of Consistent Predictions	67%	74%
Accuracy of All Predictions	56%	44%

Both models have numerous instances of inconsistent predictions, that is either forecasting a win for both teams playing in a given game or predicting a loss for both teams. In fact, the radial model experienced an inconsistent prediction in 41% of games in

the 2021 tournament (17% for the linear kernel model). However, the accuracy rate of both models increased when only considered for games in which the models made a consistent prediction, up to 74% for the radial model.

To account for this, the 2021 tournament data set was adjusted randomly to only include one observation for each game. That data frame consisted of 28 win observations and 38 loss observations. The two best final models were then run on this new data set to replicate its use in practice:

Final Results for Linear and Radial SVMs for 2021 Tournament Games:


Linear, Cost = 1						Radial, Cost = 10					
SupVect 860						SupVect 1334					
	L	W	T	Correct			L	W	T	Correct	
L	29	14	43	Incorrect	23	L	21	5	26	Incorrect	22
W	9	14	23	Total	66	W	17	23	40	Total	66
T	38	28	66	Accuracy	65%	T	38	28	66	Accuracy	67%

With the elimination of inconsistent results accuracy for both models improved, to 65% for the linear kernel and 67% for the radial kernel. *Notice that these levels of accuracy are below those from running the models with all teams in the test data, but only considering consistent results (shown above). This implies there is information to be gleaned from the inconsistent results shown in the model runs above.* The suspicion is that an inconsistent result is indicative of the teams playing being evenly matched.

So the most accurate model found to predict a winner of a game is SVM with a radial kernel and cost of 10. The model is to be run on both teams' season statistics and if a consistent prediction is given (a win for one team and a loss for the other) the model achieves 74% accuracy in the 2021 tournament. In the case of an inconsistent prediction, that same model is to be run using only one of the teams playing season statistics. This will result in a less accurate prediction, so the prudent course would be to not pick a winner if that is an option.

Conclusions

They call it gambling for a reason. This analysis explored the usefulness of 4 models in predicting the outcome of NCAA tournament games with varying success. The most accurate model in predicting 2021 tournament games (SVM) obtained an accuracy rate of 74%. While the result is certainly better than chance, it remains to be seen whether the models could be profitably used in predicting games. This is a topic for further investigation, as well as a few others noted below.



The issue here is the odds used in the gaming market which account for differences in quality of teams playing. Primarily two types of wagers can be placed on games - a "straight" wager or a "money-line" wager. In a straight wager the lesser quality team (underdog) is given "points" to be added to their total points for the game. To win a wager betting on the "higher" quality team (favorite), the favorite needs to win the game by more total points than those "given" to the underdog. A money-line wager uses straight odds, so when wagering on a favorite, more money needs to be risked than can be won. For example, risking \$150 to win \$100.

Additional work needs to be done to incorporate the odds into the models' accuracy. Money-line type wagers are consistent with the models used here, which just predict a winner or loser of a given game. But the results need to be adjusted for the odds to determine if they can be used profitably. For straight wagers, models would need to be adjusted to not only predict the winner of the game, but also the point differential.

One other topic for further investigation is the use of models in conjunction with one another. One method would be to only "wager" on games in which all models agree on the outcome. Another approach would be to combine all models into one by using machine learning to solve for weights to be given to each underlying model.

Further, recall that the variables used in all models represent regular season averages for key statistics, and are used to predict future games. Circumstances change over the course of a season - teams can improve or get worse, players can be injured or return from injury, or a team can simply get "hot" at the right time. These circumstances can change the relationships quantified in the historical data which are used to predict current games.

Overall, the first step in the process is to develop models which can correctly predict the outcomes of games. That step was examined here with reasonable accuracy. While not ready for Vegas, the models may be used to dominate your office pool.