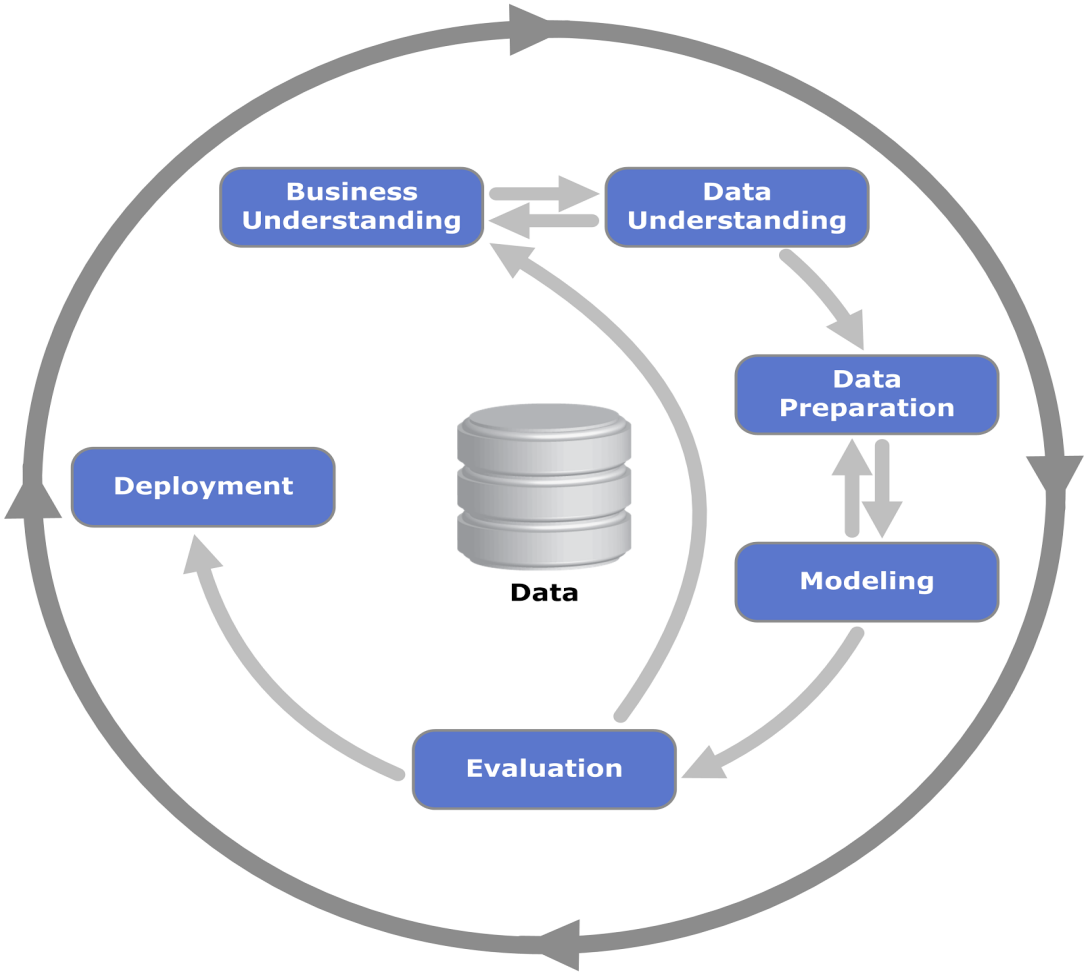
A Data-Mining Approach to Predicting the NCAA Men's Division I Basketball Tournament

Amit Balia, Naman Jain, and Julian Sy

BIA 656: Statistical Learning & Analytics
Spring 2015

CROSS-INDUSTRY STANDARD PROCESS: CRISP-DM

We approached this data mining problem using the CRISP-DM methodology. According to CRISP–DM, a given data mining project has a life cycle consisting of six phases and the phase sequence is adaptive (i.e. the next phase in the sequence often depends on the outcomes associated with the preceding phase).



CRISP-DM: The Six Phases

Business Understanding

The National Collegiate Athletic Association (NCAA) Men's Division I Basketball Tournament is a single-elimination tournament played each spring in the United States, currently featuring 68 college basketball teams, to determine the Men's Division I Basketball National Champion. The tournament was created in 1939 by the National Association of Basketball Coaches, and was the idea of Ohio State University coach Harold Olsen. Played mostly during March, it is known informally as "March Madness" or the "Big Dance", and has become one of the most popular annual sporting events in the United States. The 68 teams are divided into four regions and organized into a single elimination bracket, which pre-determines, when a team wins a game, which team it will face next.

Every year, millions of college basketball fans fill out tournament brackets to predict winners of the NCAA Men's Division I Basketball Tournament. Many submit their brackets to various contests in search of cash prizes: in 2014, Warren Buffett offered $1 billion to the owner of a perfect bracket, in which all tournament games are perfectly predicted, and $100,000 to the top 20 most accurate brackets.

In addition, millions of people place bets on individual tournament games through various sportsbooks. To make money, sportsbooks set the odds of games such that they reflect the likelihood of the outcome of the game. Sportsbooks make the most money when there is a balanced distribution of bettors betting for both teams, so they may alter the odds to encourage more bettors to pick a particular side. Bettors look to find inefficiencies in the odds, that is, odds that do not accurately reflect the likelihood of a game's outcome. In the

long run, if a bettor knows the true odds better than a sportsbook does, he is said to have "edge", and can make money.

We developed a decision support system to support either sports bettors or sportsbooks in order to:

a. help bettors make more intelligent bets on basketball games (to help bettors find "edge")
b. help sportsbooks more efficiently set odds (to help sportsbooks reduce bettors' "edge")

Thus, by using a decision support system, each party stands to make more money. To explain how a system like this can make someone money, we first have to explain "edge". In the simplest definition, edge is the long term of advantage that a bettor has over a sportsbook (or the other way around).

As an analogy, casinos are generally known to have a "house advantage", which means that, over the long run, casinos have an advantage over gamblers. This house advantage is important because it allows casinos to stay in business, while still allowing for a small percentage players to win big in the short term. The small probability of big winnings provide an incentive for players to come to casinos, even if the house advantage is always ever-present. Here, the terms "edge" and "house advantage" are interchangeable.

For sports betting, a bettor can figure out if he/she has edge by comparing a sportsbooks odds to his/her own perception of what the odds should really be. For instance, the historical odds for the Elite Eight game between Wisconsin and Arizona in the 2014 tournament was:

Wisconsin +144

Arizona -175

A positive number indicates the amount of money you might make as profit if you wager $100. A negative number indicates the amount you must wager to make a $100 profit. These are known as American odds. For the example above, this means you would have to wager $100 to make a profit of $144 on Wisconsin and you'd have to wager $175 to make a profit of $100 on Arizona.

These odds also correspond to the probability of winning, using the formulas:

$$\textit{win probability for plus odds} = \frac{100}{(\textit{plus odds}) + 100}$$

$$\textit{win probability for minus odds} = \frac{-(\textit{minus odds})}{-(\textit{minus odds}) + 100}$$

Using these formulas, the Wisconsin-Arizona odds presented above are converted to the following win probabilities:

Wisconsin: 40.9% chance of winning

Arizona: 63.6% chance of winning

Note that these percentages do not add up to 100%. In the end, sportsbooks are interested in having a balanced amount of bets for both sides, so they'll adjust the odds accordingly to provide an incentive for more bettors to bet on a particular side. Also, from the bettors' perspective, they'll set the odds slightly worse than the true odds: this is known as vigorish, which is the amount charged by a bookmaker for a bettor to make a bet.

A smart bettor might develop his own win probabilities, which would determine whether he should bet or not. For instance, he might think Arizona has a 74% chance of winning, which corresponds to American odds of -284. This means he would be willing to wager as much as $284 to make a profit of $100 by betting on Arizona. Compared to the example odds we presented in the beginning (where he would have to only pay $175 to make a profit of $100), he should see the odds presented as a deal. He would thus want to make a bet.

On the contrary, if a bettor thinks Arizona has a 55% chance of winning, this would correspond to American odds of -122, which would mean he would be willing to wager $122 to make a profit of $100 by betting on Arizona. Compared to the example odds presented in the beginning (where he would have to pay a larger amount--$175--to make a profit of $100), he should see the odds presented as "expensive", and thus he wouldn't want to make a bet.

This short primer on odds illustrates why a decision support system for predicting games can have positive business value. By providing prediction and win probability for the winner, a decision support system can provide information that is directly related to odds, which in turn is directly related to the money that is exchanged between sportsbooks and bettors.

Many data have been compiled for individual basketball teams and individual basketball games and there is a vast history of completed games we can draw insight from. This affords us an opportunity to apply a data mining approach to solving the problem of sports prediction.

Data Understanding

We used data from two main sources (Kaggle and KenPom.com), which served as our raw data. We combined data from both raw sources to create our final dataset, which is described in the Data Preparation section. Here, a brief overview of the raw data is presented.

The data in the Kaggle dataset is comprised of the results of every single regular season game in the 2014 season. Every instance of the dataset represents the data for a single game. There was a total of 5362 regular season games, so this dataset has 5362 total rows. In addition to the two teams that played, and the winner of the game, the dataset provides various box score data (e.g. free throws made, steals, etc.) for each team. There are no missing values. The relevant attributes from the dataset are presented in the table below. Note that all variables are numerical.

Kaggle Data Attributes

Attribute Name	Attribute Description
wteam	unique ID of winning team
lteam	unique ID of losing team
wfgm	winning team's field goals made
wfga	winning team's field goals attempted
wfgm3	winning team's 3 pt field goals made

wfga3	winning team's 3 pt field goals attempted
wftm	winning team's free throws made
wfta	winning team's free throw attempts
wor	winning team's offensive rebounds
wdr	winning team's defensive rebounds
wast	winning team's assists
wto	winning team's turnovers
wstl	winning team's steals
wblk	winning team's blocks
wpf	winning team's personal fouls
lfgm	losing team's field goals made
lfga	losing team's field goals attempted
lfgm3	losing team's 3 pt field goals made
lfga3	losing team's 3 pt field goals attempted
lftm	losing team's free throws made
lfta	losing team's free throw attempts
lor	losing team's offensive rebounds
ldr	losing team's defensive rebounds
last	losing team's assists
lto	losing team's turnovers
lstl	losing team's steals
lblk	losing team's blocks
lpf	losing team's personal fouls

The KenPom dataset is comprised of various metrics for every single team (metrics which aren't available in the Kaggle dataset). Each instance consists of a single team, and various

metrics for that team that have been averaged on a per-game basis over the entire regular season. Since there are 351 total college basketball teams, this dataset has 351 rows. There are no missing values. The relevant attributes from the dataset are shown below. We set the team_id in this dataset to the corresponding team_id in the Kaggle dataset, which allowed us to do the requisite SQL-esque join operations during the data preparation. Note that all variables are numerical. When KenPom provides an "adjusted" metric, it means they've adjusted it based on a team's strength of schedule.

KenPom Data Attributes

Attribute Name	Attribute Description
team_id	unique ID of the team
AdjTempo	adjusted tempo: possessions per game, adjusted for opponent
AdjOE	adjusted offensive efficiency: points scored per 100 possessions, adjusted for opponent
AdjDE	adjusted defensive efficiency: points allowed per 100 possessions, adjusted for opponent

These two datasets allowed us to build a dataset with all regular season games, with the winner as a target, and the average per-game metrics for each team as the independent variables. This is explained further in the Data Preparation phase.

Data Preparation

During the data preparation, our goal was to build a single dataset from the Kaggle and KenPom datasets, with each instance representing a single game, and each instance having metrics for the two teams playing in that game. The metrics would represent the averaged

value of that metric on a per-game basis over the regular season. For instance, for the attribute *steals*, this would be the average steals per game for a team.

Since the Kaggle data only has a team's metrics for a particular game, we used Excel to derive the per-game average of each metric for every team. The resulting dataset was a dataset comprised of teams and their per-game average metrics. Then, using Python's pandas library, we inner joined this dataset with the KenPom dataset. The final result was a dataset of teams, and their metrics (from both KenPom and Kaggle data), averaged per-game. We will refer to this data set as the *team dataset*.

To build the dataset of games, we took the original Kaggle dataset of games, and executed two inner joins with the the teams dataset. The first inner join gave us a dataset that had all the games and only one of the team's average metrics. The second inner join allowed us to create a dataset that also had the other team's average metrics. We set this up so that there would a "Team A" and a "Team B", with our target variable (winner) being denoted as "A" or "B".

At this point, we realized the target variable was entirely comprised as "A", so we ran a randomization code that divided the target variable as both "A" and "B" in roughly a 50/50 split.

Finally, we took this dataset and derived new metrics, by executing operations between various columns. For instance, we were interested in the difference between the teams' respective values for *field goals made*, so we subtracted team A's *field goals made* with team

B's *field goals made*. A full list of relevant attributes for the final dataset are shown in the table below. Note that for all attributes except *winner*, the values are numerical. Also, there was no missing data in this dataset. Our models were built off of this dataset.

Attributes of the Final Training Set

Attribute Name	Attribute Description
teamA	unique ID for team A
teamB	unique ID for team B
winner	the winner of the game, 'A' or 'B'
fgm_diff	the difference in average field goals made between team A and team B
fga_diff	the difference in average field goals attempted between team A and team B
fgm3_diff	the difference in 3-point field goals made between team A and team B
fga3_diff	the difference in 3-point field goals attempted between team A and team B
ftm_diff	the difference in free throws made between team A and team B
fta_diff	the difference in free throws attempted between team A and team B
or_diff	the difference in offensive rebounds between team A and team B
dr_diff	the difference in defensive rebounds between team A and team B
ast_diff	the difference in assists between team A and team B
pf_diff	the difference in personal fouls between team A and team B
tempo_diff	the difference in adjusted tempo between team A and team B

oe_de_diff	the difference between team A's adjusted offensive efficiency and team B's adjusted defensive efficiency
de_oe_diff	the difference between team A's adjusted defensive efficiency and team B's adjusted offensive efficiency

In addition to this dataset, we built a test set for all tournament games. This was done by looking through ESPN.com to find the tournament matchups, and manually inputting the matchups into a spreadsheet. Then, using the same join operations described earlier, we were able to add the average metrics for each team. This data was not used for model building, and was only used to evaluate the performance of our final model.

Modeling

Since the overall business problem from phase 1 was to predict 2014 NCAA Tournament, the team chose to apply the following techniques: (1) Logistic Regression, (2) Decision Tree Classifier (CART), (3) Support Vector Machines, (4) Gaussian Naive Bayes, and (5) Random Forest Classifier. We took the data from the Data Preparation phase and divided the data into 70-30 ratio, where 70% of the data is used to train the model and 30% to validate the model.

A brief summary of the techniques we tried and our reasoning for including them:

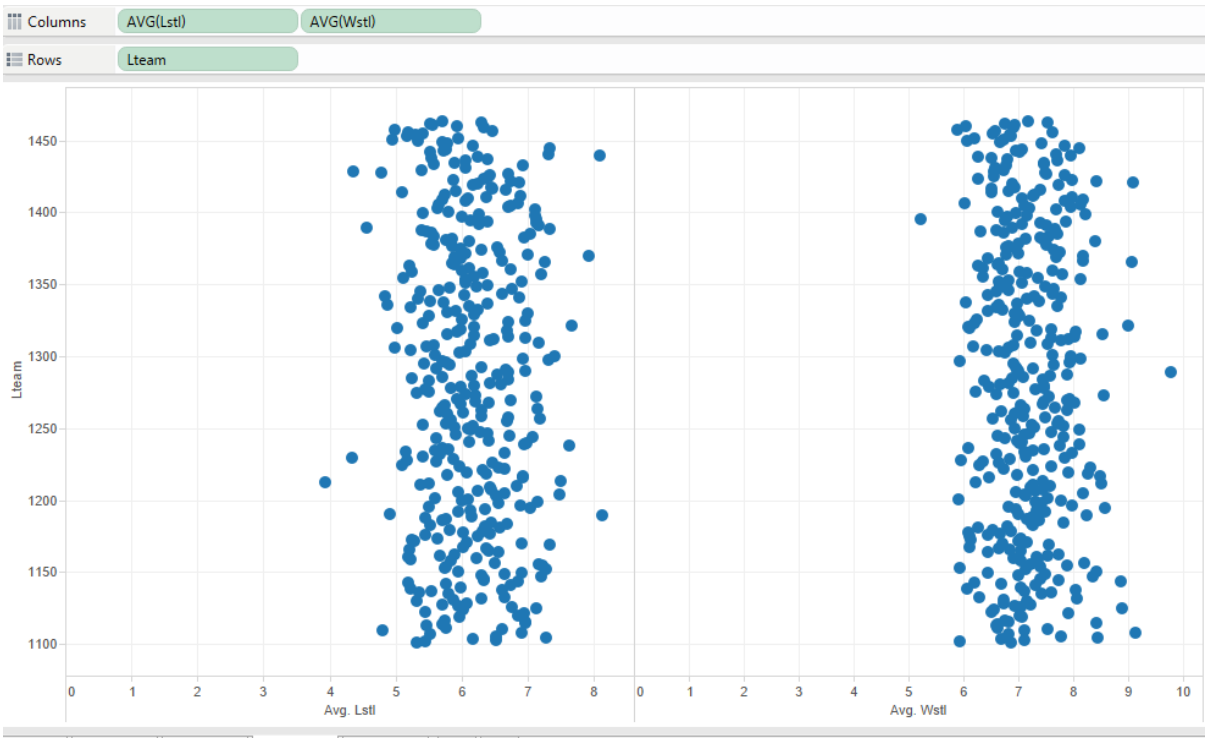
Logistic Regression: We choose this because it naturally provides a probability output when classifying data, and it is known to serve as a benchmark with which other algorithms are compared.

Decision Tree (CART): Decision trees are good for interpretability and complex data. Also, we felt we had a large dataset, and decision trees are known to work well with large datasets. In practice, we wouldn't use this algorithm for deployment, because it doesn't give us a probability output, but we wanted to compare the performance of a single decision tree with that of a bagged approach (random forests).

Support Vector Machine (SVM): In case of non-linearity, we wanted to take advantage of the kernel trick of SVM. The disadvantage is that SVM is not naturally probabilistic; however, we can obtain probabilities through Platt scaling, even if it is somewhat expensive to do.

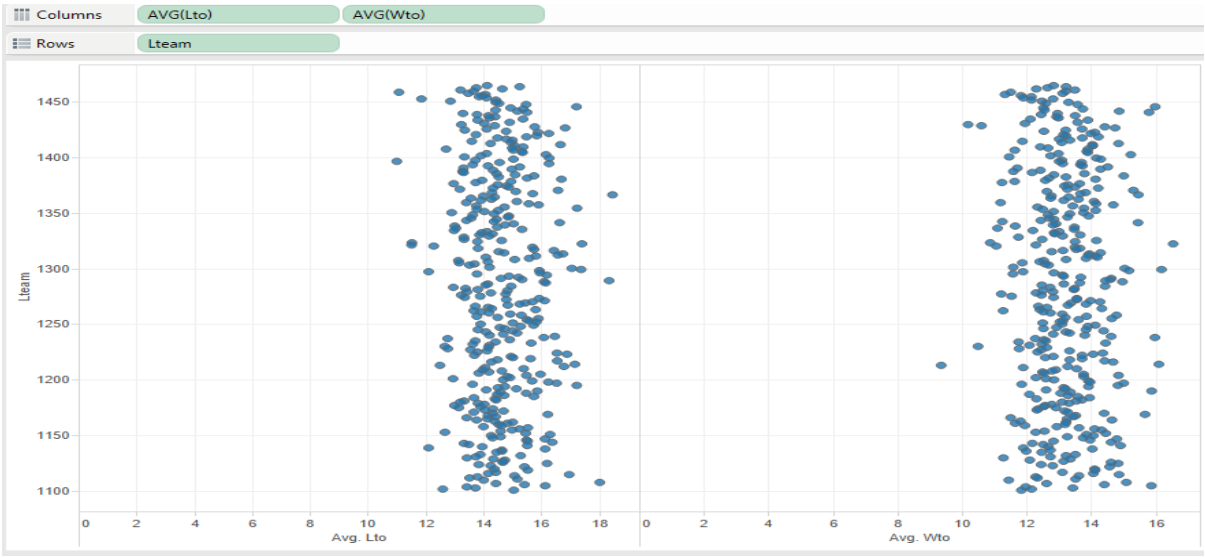
We also used data visualization techniques using the software Tableau to explore the importance of certain variables and also to determine what impact certain variables have on the target variable in addition to doing it in Python. Below are few visualization examples on how certain basketball metrics are important in the winning a particular basketball match.

Eg 1: Impact of Average number of 'steals' (stl) for a team in a match



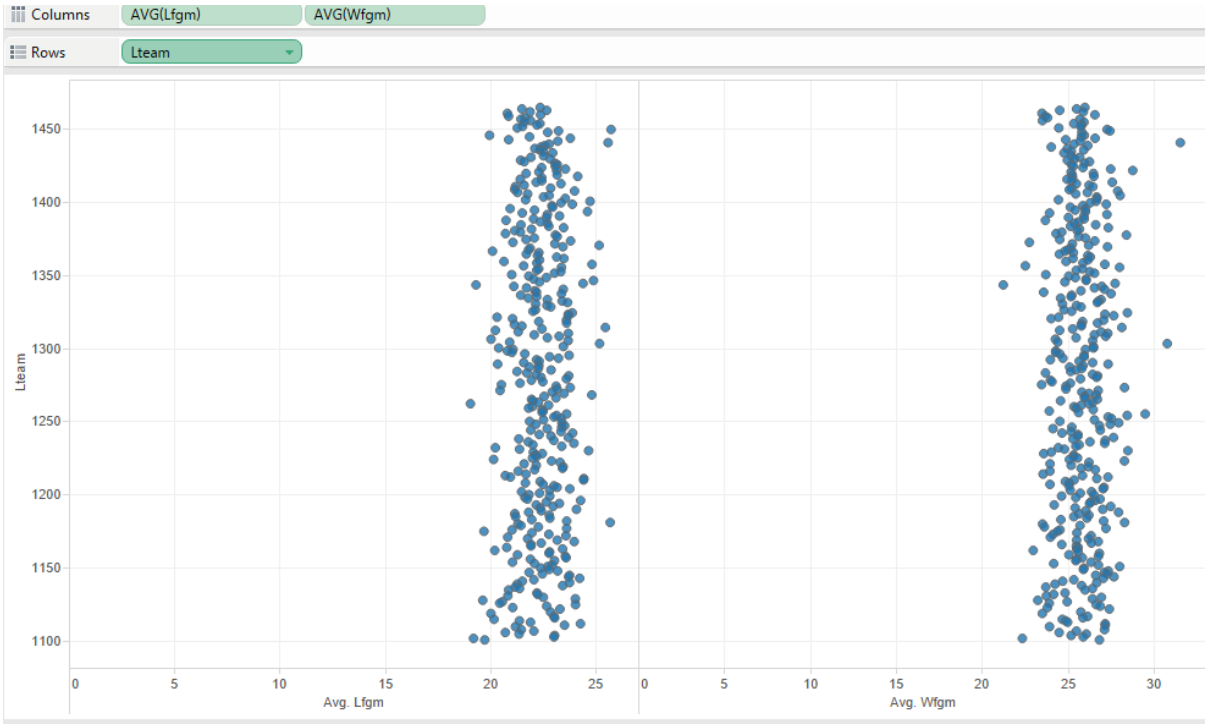
Higher number of steals in a match for a team indicates better performance and thus a better chance of winning a game. In the above example, on the x-axis we have the average number of steals for a team when they lose on the left and the average number of steals for a team when they win on the right. The Team IDs are on the y-axis. The above visualization clearly shows the winning team in a match has higher number of steals (the blue dot is above 7 on average for the winning team and below 7 for the losing team).

Eg 2: Impact of Average Turnovers (‘to’) for a team in a match



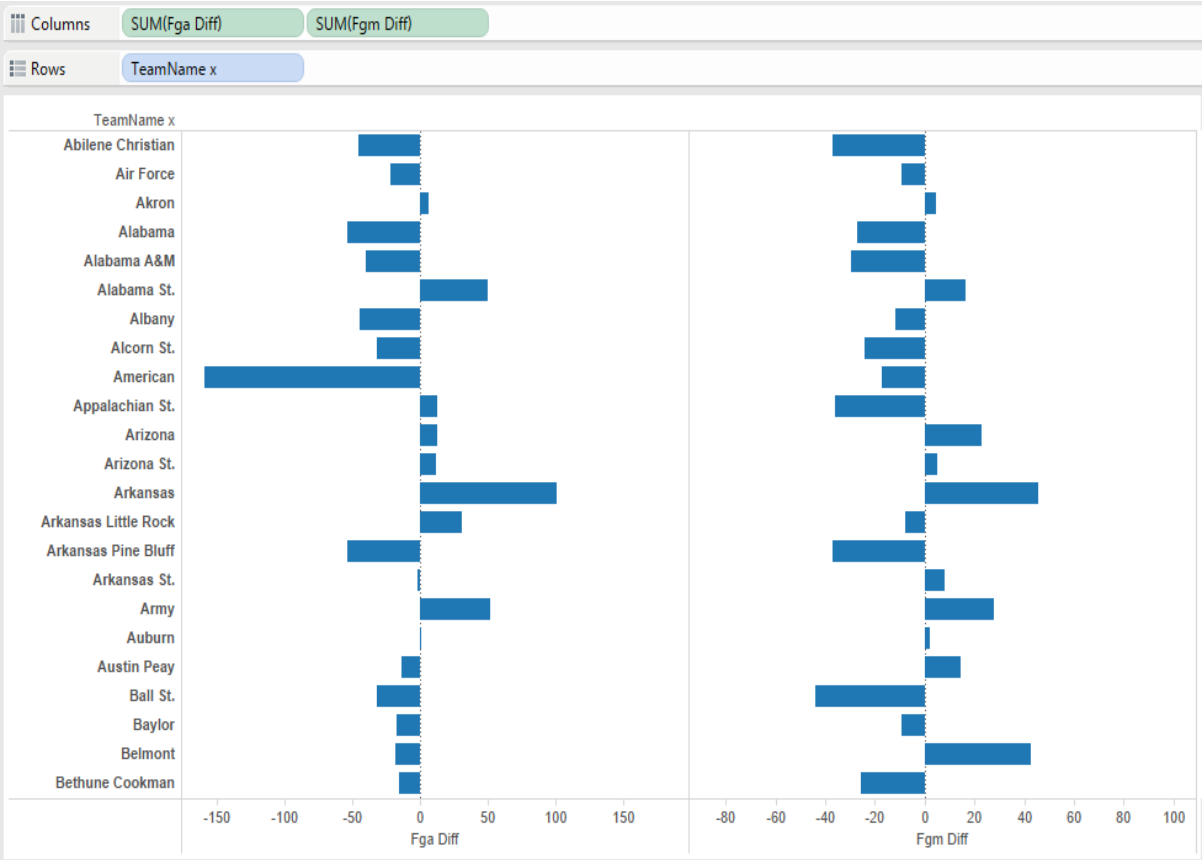
The average number of turnovers in a match are indirectly proportional to a team's chances of winning a match. In the above example we have the average number of turnovers for a losing team and the average number of turnovers for a winning team as columns. And we have the Team ID as rows. The above visualization clearly shows the winning team in a match has lower number of turnovers per game (the blue dot is in the range of 14-18 for the losing team and in the range of 12-15 for the winning team).

Eg 3: Impact of Field goals made (‘fgm’) for team in a match



The average number of field goals made in a match are directly proportional to a team’s chances of winning a match. In the above example we have the Avg number of Field goals made for a losing team and the average number of Field goals made for a winning team as columns. And we have the Team ID as rows. The above visualization clearly shows the winning team in a match has higher number of Field goals made per game (the blue dot is in the range of 20-24 for the losing team and in the range of 24-28 for the winning team).

Eg 4: Difference of Field goals attempted and field goals made for a team



The above visualization shows the comparison of the differences between the two competing team's field goals attempts and the field goals made. It shows importance of finding out this difference and hence the final training set contains this variable which will help us in predicting the target variable which is the winner of a particular match.

Evaluation Phase:

In this phase the evaluation of all the above mentioned models was done, and finally the best performing model was chosen based on its performance on the validation set. The performance of models developed in the modeling phase are as follows:

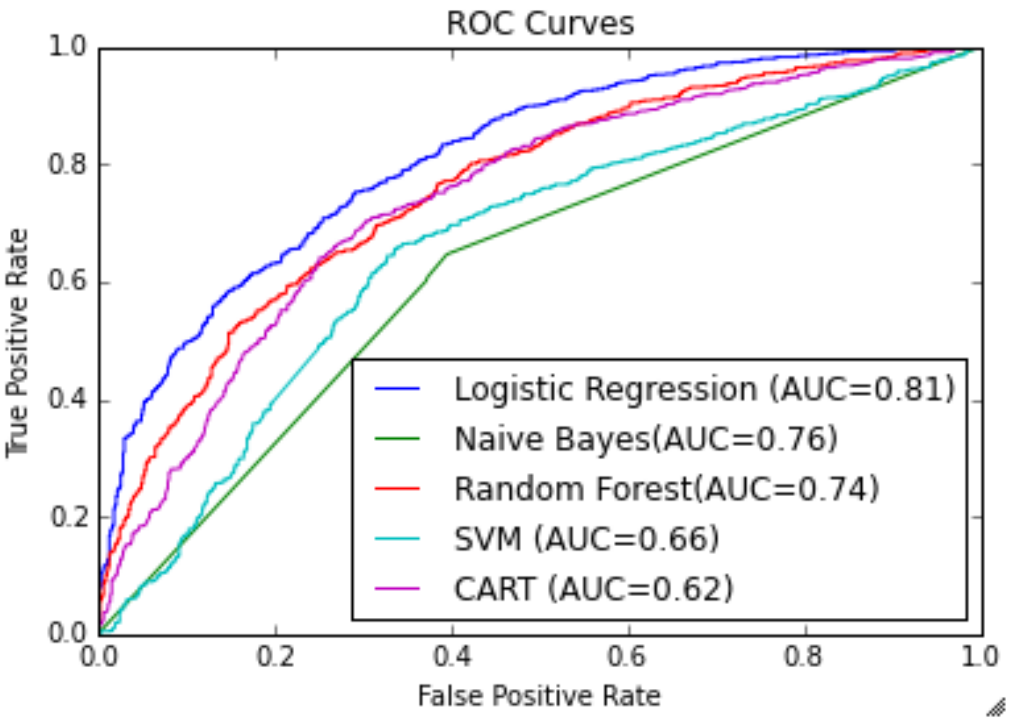
Logistic Regression gave AUC of 0.81.

Naive Bayes gave the AUC of 0.76.

Random Forest gave the AUC of 0.74.

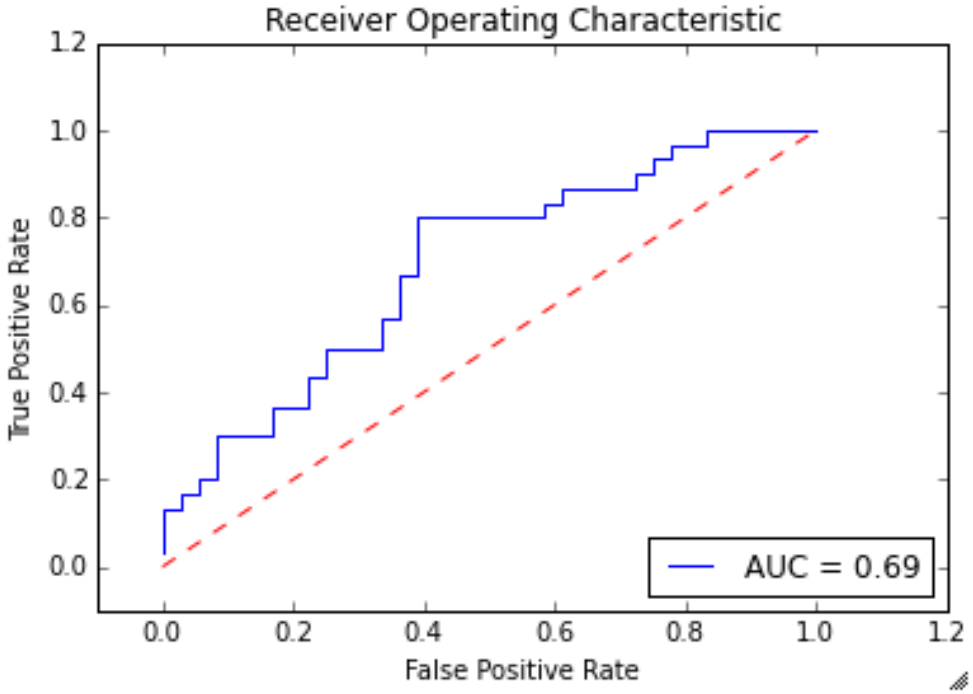
SVMs gave the AUC of 0.66.

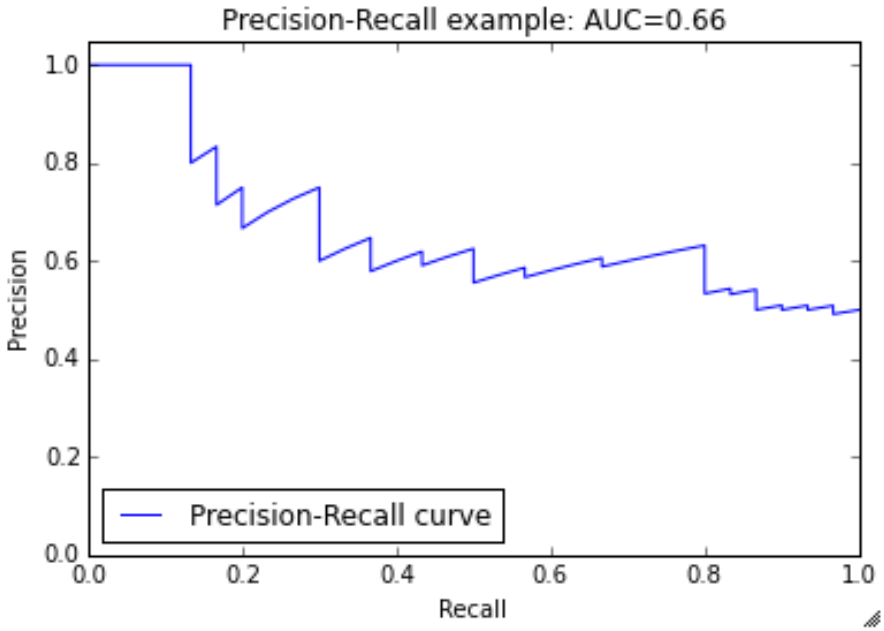
CART gave AUC of 0.62.

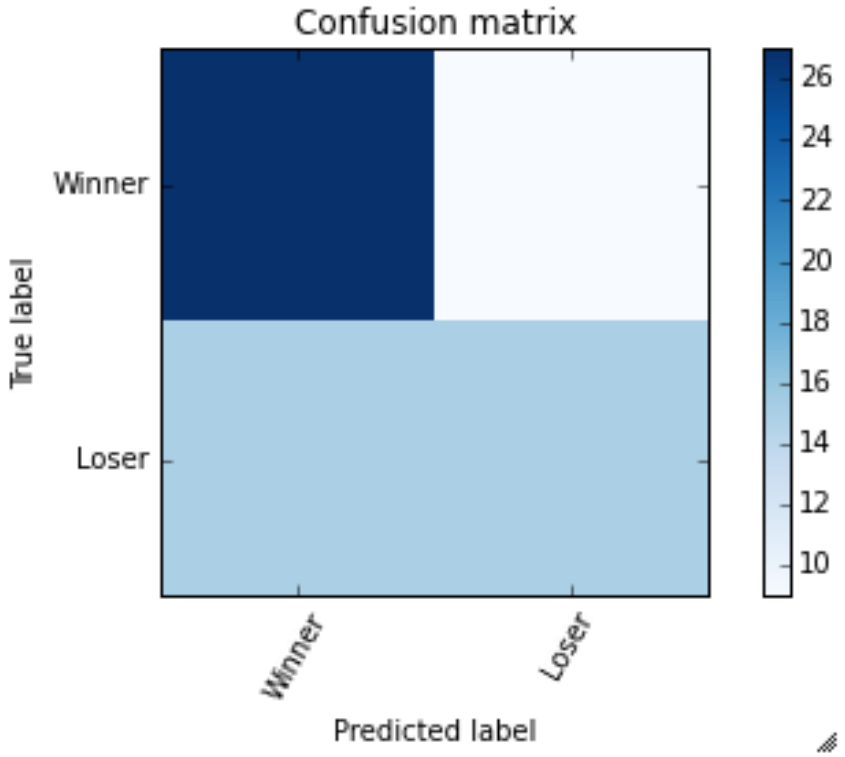


Based on the ROC curves, we chose Logistic Regression as the model we would test on the final test set as it had the highest area under the curve (AUC) of 0.81. When running our

model on the final test set, we obtained a classification accuracy of 0.64 and an AUC of 0.69. The results when running our model on the final test test are also shown in the graphs below.







In practice, bettors will choose to make a bet depending on the amount of edge he feels he has over the sportsbook. Therefore, when evaluating whether the data model is useful, a bettor needs to only check how often he is right *when he decides to bet*. If he's wrong about the outcome of a game when he doesn't decide to bet, it doesn't matter since he hasn't laid any money down.

In the end, to determine whether using our decision support system was effective, a bettor or a sportsbook should compare their long-run profits from before they started using our system to their long-run profits after using our system. If it increases, this indicates that one of them was able to raise their edge over the other. If no improvement is found, then bettors and sportsbooks should move on to another strategy.

Deployment

After comparing models in the evaluation stage, we deployed the highest performing model into a small web application, which allows users to predict the results of any matchup between any two Division I basketball teams. The web application was developed in Python, using the Flask web application framework, the Jinja2 template engine, and Bootstrap for the front-end.

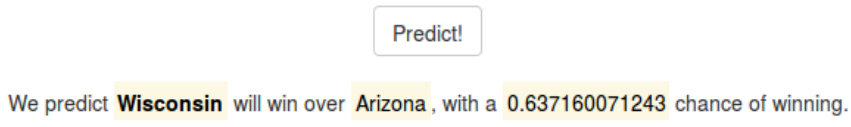
The application is currently being hosted on the Heroku cloud platform, and can be found here: http://bia656-ncaa-prediction.herokuapp.com/.

The code is available here: https://github.com/syju/bia656-ncaa-prediction-web

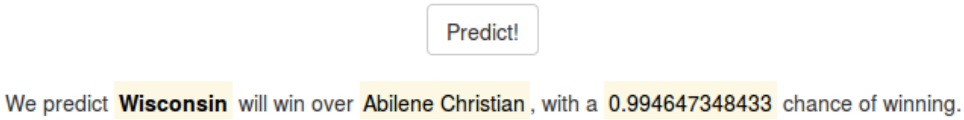
The home page of the web app is shown below. Users can select two teams from the dropdown menu, and can click 'Predict!' to obtain the prediction.

Web application home page.

An example prediction is shown in below. Both the winner of the matchup and the probability of winning is shown. When comparing the probability output with what we would expect, the probability output is sensible. When comparing two teams ranked in the top 10 of the USA Today Coaches Poll (Wisconsin and Arizona), the probability of Wisconsin winning is 63%. When comparing a top 10 program (Wisconsin) with a team that's largely unknown in the basketball world (Abilene Christian), the probability of Wisconsin winning is 99%. What this shows is that we can be more certain that big name/top-tier programs will beat schools that aren't known for basketball, and that we're less certain when we compare teams that are very similar (i.e. are both top-ranked teams).



Prediction of matchup between Wisconsin and Arizona.



Prediction of matchup between Wisconsin and Abilene Christian.

Because this system depends on a model built from data in the entire regular season, a caveat of this system is that it is designed to be used only in the post-season (i.e. during the NCAA tournament). In addition, since the model was developed using only data from a single season (2014), it is designed to be used only for predicting games in the 2014 season. The model is static, so as the tournament progresses, it does not update the model to take into account results from tournament games.

A word of caution: this system's predictions and probability estimates should not be the be-all-end-all system for sports bettors and sports books. Instead, this tool should be used as one of many tools, which as a whole can probably provide a more accurate and reliable picture about basketball games than this system can provide alone. In particular, we expect sports bettors and sports books to have significant domain expertise, so when their beliefs and our system clash, we encourage them to take our results with a grain of salt. This approach to predicting basketball games can mitigate risk, and can prevent users from making unusual and potentially money-losing decisions.

In the future, we expect to update the model yearly, so that it can be used to predict games for the future tournaments.