APPLICATIONS OF BALANCE OPTIMIZATION SUBSET SELECTION

BY

SHOUVIK DUTTA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Industrial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Adviser:

Professor Sheldon H. Jacobson

Abstract

Balance Optimization Subset Selection (BOSS) is a framework designed to be used for causal inference on observational data. The theoretical foundation for the BOSS framework has been provided in the literature; this thesis aims to provide some examples of the practical value of BOSS by using it on two problems. The first application is using BOSS to determine a subset of users who would be suitable targets for marketing efforts, and the second application is using BOSS to identify potential first-round upsets in the NCAA basketball tournament. Finally, this thesis delves into another area of college basketball and attempts to model the process of the NCAA tournament selection committee using a decision tree.

Acknowledgements

I would first like to thank my adviser, Professor Sheldon H. Jacobson, for pushing me to accomplish more than I thought I would, always making time for my numerous questions, and tolerating my many foibles. Also, many thanks to Dr. Jason Sauppe, who both provided suggestions that greatly increased the strength of the research, and helped me with the writing of the resultant papers. Additionally, I appreciate the generosity of the National Science Foundation [SES-0849223], who partially funded this work. Finally, my sincerest gratitude to my parents and my brother, who provided constant encouragement throughout my degree-earning process.

Table of Contents

1 Introduction

Balance Optimization Subset Selection (BOSS), was originally introduced in Nikolaev et al., 2013 as a way to find comparable groups to use as treatment and control groups in observational studies. The goal of BOSS is to find a small group of units out of a large pool that is similar to a target group. To do this, we define covariates that characterize the units in each group. A covariate is a factor that provides some information about the group. For example, one characterization of a group of people might include a combination of age, profession, and gender. Since BOSS operates on groups, we look at the distribution of each covariate across groups. For example, if the covariate in question is the age of people in a group, the values of the covariate would be how many people in the group are each age. The distribution of that covariate would consist of the proportion of people in the group who are each age. We are looking to find a group of users who have the same distribution for each of these covariates. Let us define the group we would like to match as the *treatment group*, the pool out of which we would like to select the new group as the *control pool*, and the group we select as the *control group*. The objective of BOSS is to select a control group that has covariate distributions as similar as possible to those of the treatment group.

BOSS was designed as a framework to solve the problem of drawing conclusions from observational data when randomized experiments are impractical or impossible. Examples of such cases would include testing the long term effects of smoking or high radiation exposure, where it would be unethical to force people to smoke or expose them to high levels of radiation. Instead, affected people who have already been exposed to these effects (whether by choice or accident) can be observed. This data, known as *observational data*, is so named because the data is gained by observing (rather than creating) it. Observational data has many advantages - it is often easier to obtain because it uses existing data rather than creating new data (via experiments), and it can be often obtained in larger quantities. However, the important difference between experimental and observational data is that experiments are carried out using random assignment. When conducting an experiment, each unit is randomly assigned to either the treatment or the control group. This ensures that the distribution of the covariates (attributes) in both groups are (stochastically) identical, which removes any bias and isolates the effect of the treatment. In observational data, there may be differences between the treated and control units, such as smokers being generally older than non-smokers, or smokers being less physically active. These differences can bias the treatment effect with a shift caused by another difference between the treatment and control groups. If the people in the treatment group lead less active lives than the people in the control group, any difference in long term health may be caused by smoking or by the difference in lifestyles. Drawing a strong conclusion necessitates the removal of these potential biases.

Removing bias in the data is further complicated by the presence of both observed and unobserved

covariates. Observed covariates are those that are present in the data set, whereas unobserved covariates are those not explicitly given. Unobserved covariates present a challenge, since dealing with them requires information not present in the data set. Observed covariates can be handled by operating on the data to produce treatment and control groups with equivalent distributions of covariates according to a given balance measure.

Nikolaev et al., 2013 proved that having identical marginal distributions for each covariate between the treatment and control groups will minimize the bias between the groups under a separable response function. Rather than finding a corresponding unit for each unit in the treatment group, BOSS optimizes an overall balance measure. This balance measure can be defined as the difference between the distributions of each coviarate of the two groups. Other possible measures include the difference in the sums of unit values in each group or various statistical test functions. The range of values that each covariate can take is partitioned into a set of buckets, and the distribution for that covariate is formed by the proportion of users who have a value for that covariate in each bucket. BOSS attempts to ensure that each bucket for each covariate contains the same proportion of users in both the treatment and control groups.

The theoretical foundation for BOSS has been laid out in the literature. This thesis aims to apply BOSS to real problems in order to show its efficacy and practicality. This allows us to see what kind of results we can achieve using BOSS, as well as how BOSS can be incorporated into a multi-step process containing standard machine learning methods. Here we apply BOSS to two problems. First, we apply it to targeted marketing, where we aim to select a subset of potential users who would be suitable marketing targets. Specifically, we use Netflix data and attempt to select users to whom we should advertise a given target movie. We do this by finding similar movies to the target movie and seeing how users who liked the target movie rated these similar movies. These ratings are then used as covariates for BOSS to select the potential marketing targets. We do this for each of 2000 target movies and measure the improvement our results provide over random selection. The second application is identifying upsets in the NCAA basketball tournament, where we try to identify potential upsets in the first round of the tournament. We aggregated 115 different statistics for each team over each season, identified the 15 most important, and used these to build historical profiles for what statistical matchups have contributed an upset. Then, an ensemble of BOSS models is run on different subsets of these statistics and combined to produce two potential upsets for each year between 2003 and 2015.

The final chapter of this thesis attempts to replicate the decision-making process of the NCAA basketball tournament selection committee using a decision tree. Each year the committee selects teams to invite to the tournament, but the exact method they use to rank the teams is not public. We build a decision tree that, given any two teams, is able to determine the stronger team following a pattern consistent with the

selection committee's historical choices. While this is not related to BOSS, it is a tangential project spun off from the prior basketball work done with BOSS.

2 Targeted Marketing

2.1 Introduction

The increasing prevalence of online retailers and media providers presents customers with a diverse set of options to choose from when buying or consuming products. When faced with too many choices, customers often have neither the time nor the inclination to manually sift through them all to find the most interesting items. The ability of retailers to reduce the work of the customer by recommending products to users based on the specific taste of each individual presents an opportunity to greatly increase user satisfaction. The software to do this, known as a recommender system, has become an increasingly large part of the user experience on sites such as Netflix. As users on these sites consume goods and leave feedback (typically in the form of numeric ratings), their behavior can be analyzed to create a profile to quantify their interests, which is then used to provide predictions on what the user will or will not like in the future. In the case of Netflix, these behaviors include rating a movie, stopping a movie early, or even viewing a movie page but choosing not to watch it. All of these factors can be used to build a profile of a particular user.

Existing research (Lü et al., 2012), has focused on predicting exactly what a user will rate a particular item. These methods typically revolve around building a model utilizing all past user data. This paper presents an alternative methodology to select a relatively small group of users out of a large pool who are expected to like a given product. The applications for this may include a marketing campaign, where a company has an extremely large database of users and wants to advertise a product only to those customers who are likely to purchase and enjoy it. Rather than predict every users rating for the product, we operate on the group level, choosing a group of a specific size to market to. In order to do this, we characterize the group of users who are known to like the product (based on past ratings or sales history) and find a group of users who (as a group) share the same characteristics. This method is particularly interesting because it requires only a small number of factors to provide a suitable set of users, which means that it can potentially be used even when only a small amount of data is available. While traditional methods require computing ratings based on the entire history of a user, this method uses a small number of user attributes. The methodology was tested in various configurations using the Netflix Prize dataset, where we select users who are similar to others who have liked a particular movie and who we therefore theorize are disposed to like the movie, and compare them against how they actually rated the movie.

2.2 Background

Attempting to predict user ratings is a problem that has been studied in the literature (Lü et al., 2012), especially in the domain of internet commerce (Lee, Liu, and Lu, 2002). Since perfect user-item rating predictions would allow businesses to know what each customer would like with certainty and hence allow them to market having perfect information, companies and researchers have experimented to try and get as close as possible to perfect prediction. There are a variety of different methods for doing so, including Exact Matrix Completion (Candès and Recht, 2009), which attempts to predict the rating of every item by every user simultaneously, and Content/Collaborative Filtering (Koren, Bell, and Volinsky, 2009; Cheung et al., 2003), which attempts to find similarities between items and users in order to to predict future ratings.

2.2.1 Exact Matrix Completion

Exact Matrix Completion attempts to predict the rating of each product by each user given existing user-item ratings by creating a completed matrix where the missing user-item ratings are given values. This method uses a matrix M where each row represents a user in the set of users U, and each column represents a product in the set of products P. The cells M_{ij} in the matrix are the ratings for the product by the user. A general assumption used is that the matrix will be of low rank (Candès and Recht, 2009), since a user's tastes will likely depend on only a few factors relative to the number of users and products. The objective therefore is to find a low-rank matrix which matches the existing user ratings exactly but also fills in unknown ratings. This problem can be formulated as

$$\min \{\text{rank}(X)\} \tag{1}$$

subject to

$$X_{ij} = M_{ij} \text{ for all } i \in U,\, j \in P \tag{2}$$

where X is the completed matrix and M is the provided matrix. The range for values of X will be the range of potential ratings. If there exists a low-rank matrix which fits the input data, this method would recover it. However, this optimization problem is NP-hard (Candès and Recht, 2009), and hence is impractical to use on large datasets.

One way to reduce the time required to solve the optimization problem is to observe that if a matrix has rank r, then it has r nonzero singular values. We can thus modify the problem to minimize the sum of the singular values (known as the nuclear norm) rather than minimize the rank directly. The nuclear norm is defined as

$$||X||_* = \sum_{k=1}^{n} \sigma_k(X) \tag{3}$$

where $\sigma_k(X)$ is the kth largest of the n singular values of X. The optimization problem is then defined as

$$\min \{||X||_*\} \tag{4}$$

subject to

$$X_{ij} = M_{ij} \text{ for all } i \in U, j \in P \tag{5}$$

This formulation can be optimized using semidefinite programming as described in Candès and Recht, 2009.

Exact Matrix Completion has several drawbacks. First, it is not always possible to recover the completed matrix from a sample. If a particular movie has not been rated, then there is no way to accurately complete ratings for that movie, despite the fact that ratings may be predictable by relating the movie to other similar movies (as done by other methods). Furthermore, the lowest rank matrix may not be the best representation of the true ratings. Finally, the matrix must be reevaluated completely upon the introduction of new users, items, or new ratings if they do not fit the previously predicted matrix. This can be extremely computationally intensive, and hence, not suitable for larger systems.

2.2.2 Content and Collaborative Filtering

A different approach is to focus on building relationships between users or items. One example of this is *content filtering*, which uses a pre-built profile of each user or item to discover similarities. A user profile could include factors such as age, gender, or profession as well as answers to questions posed during profile creation. An item profile would consist of product attributes - in the Netflix case, movie profiles could have lead actors, director, and genre. These profiles could then be used to build relationships between users or items by identifying similar users and similar movies. However, creating these profiles can be time-consuming and might require additional information not immediately available to the retailer (such as demographic information or expanded movie attributes), and hence may result in incomplete or inaccurate profiles. While this method has the advantage of relying on more data rather than attempting to extract information from user behavior, the initial cost of information is often prohibitive, since users are apt to leave the site and use one that appears less intrusive. An ideal system would require no additional input from the user and a minimal amount of information from the company, and would learn the relevant connections

itself.

Collaborative Filtering is a recommendation system based solely on past behavior which mitigates some of the downsides of content filtering that was developed by Xerox for their Tapestry product in Goldberg et al., 1992. Collaborative Filtering relies solely on past user behavior such as previous purchases, ratings, or even page views that did not lead to a purchase, and does not require the creation of profiles, which can be an asset when creation of such profiles is difficult. The main advantage of Collaborative Filtering is that it is applicable to systems containing any type of product since it does not rely on the inherent characteristics of the product being recommended. It can also find hidden relationships that would be difficult to uncover via content filtering, since the patterns would be learned from data rather than manually entered. The drawback is that since it relies on past behavior, collaborative filtering is unsuitable for making predictions for new users or products, since no information is available about them. Content Filtering is more suitable for such cases, since the creation of profiles provides initial information that can be used.

Collaborative Filtering can be broken down into two forms (Koren, Bell, and Volinsky, 2009): *neighborhood methods* and *latent factor models*. Neighborhood methods find relationships between items or between users, and assign to each pair a similarity score to express how similar the items or users are. *Neighboring* items or users are the items or users most similar to a given item or user. For example, a user's neighbors would consist of the users who rate items most similarly to the given user. The item-oriented method might predict the rating of an item by a user based on how the user rated neighboring items, while the user-oriented method predicts the rating by how neighboring users rated the item.

Finding neighboring items or users is done using the K-Nearest-Neighbors algorithm (Koren, Bell, and Volinsky, 2009), where for each item or user the k most similar items or users are found. In the users case, the k most similar users are found, and then a predicted rating is generated by finding the average rating given by the similar users. Alternatively, the rating can be generated using a weighted average, where the weight is the similarity between the target user and the similar user. Also, because some users tend to systematically rate higher than others, the average rating of each user should be incorporated, in which case the predicted rating $\widetilde{r}_{ui}$ by user u for product i would be

$$\widetilde{r}_{ui} = \bar{r}_u + k \sum_{v \in \widehat{N}_u} s_{uv}(r_{vi} - \bar{r}_v) \tag{6}$$

where $\hat{N}_u$ is the set of users similar to user u found using the K-Nearest-Neighbors method, s_{uv} is the similarity score between user $u \in U$ and user $v \in \hat{U}_u$, and $k = 1/\sum_v |s_{uv}|$ is the normalization factor to keep the rating scale the same. The advantage of the weighted average is that the similarity between users is taken into account, since a more similar user is more likely to be an accurate predictor of the target user's

rating. The same technique can be applied to items instead of users, where k similar items would be found and then the user's ratings of the similar items would be averaged, optionally including the similarity scores between the items as weights and the average item rating.

In order to use the K-Nearest-Neighbor method, a metric for measuring the similarity between users or items must be defined. In the case of explicit ratings (such as the Netflix dataset), there are several different metrics that are used (Xu and Tian, 2015). One example is the Cosine index, which defines the similarity as

$$s_{uv}^{\cos} = \frac{\boldsymbol{r}_u \cdot \boldsymbol{r}_v}{||\boldsymbol{r}_u||||\boldsymbol{r}_v||} \tag{7}$$

where $\boldsymbol{r}_u$ and $\boldsymbol{r}_v$ are rating vectors for user u and user v. Another metric is the Pearson coefficient, defined as

$$s_{uv}^{PC} = \frac{\sum_{i \in O_{uy}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in O_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in O_{uv}} (r_{vi} - \bar{r}_v)^2}} \tag{8}$$

where O_{uv} is the set of items rated by both users u and v, r_{ui} is the rating of item i by user u, and $\bar{r}_u$ is the mean of all ratings given by user u.

The other form of collaborative filtering, latent factor models, attempts to characterize both items and users to find patterns to explain existing ratings. The goal is to identify such patterns and determine a predictive rating by looking at the correspondence between the factors of a given user and item. These factors are similar to what may be produced in the content filtering approach, except that these would be discovered by the model. Discovered factors of movies may include genre, violent content, kid-friendly content, visual style, or factors that are not easily described. A movie would be represented as a vector of values for each factor, where a high value means the movie exhibits that factor to a high degree. For users, the value for a factor would measure how much that user likes movies with that factor (a high score for a factor would indicate that a user likes movies that exhibit that factor). The exact ranges of the values would depend on the rating system being used. Since the movies and users are being mapped to the same space, the predilection of a user to like a movie would be calculated as the dot product of the user and movie vectors.

Matrix Factorization (Koren, Bell, and Volinsky, 2009) separates the ratings into the product of the user and item factors. Using the same input format where the rows of a matrix represent users, the columns represent items, and the values represent user-item ratings, matrix factorization creates one matrix of users and one matrix of items, where the columns in each matrix are the values for the different factors learned by the model. To do this, let the dimensionality of the factors be f. Each item i is then represented by a vector $q_i \in \mathbb{R}^f$ and each user u is represented by a vector $p_u \in \mathbb{R}^f$. The elements of q_i represent the extent to which

item i prefers each factor. The elements of p_u measure the extent to which user u possesses each factor. The dot product $q_i^T p_u$ measures the correspondence between item i and user u, with higher correspondence representing a user being more interested in the product. The predicted rating of item i by user u is thus

$$\hat{r}_{ui} = q_i^T p_u \tag{9}$$

The goal is to determine how to map each item and user to the factors such that the predicted ratings r_{ui} match known ratings as closely as possible. This is accomplished by minimizing the regularized squared error between the projected ratings and the actual ratings, while simultaneously avoiding overfitting. The latter objective is done by penalizing the model based on the magnitudes of p_u and q_i, which disincentivizes the model from trying to exactly fit the training data at the expense of generalization. The minimization problem is given by

$$\min_{q^*,p^*} \left\{ \sum_{u,i \in K} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2 \right\} \text{where } p_i, q_i \in \mathbb{R}^f \forall i \tag{10}$$

where K is the set of user-item pairs with known ratings (Koren, Bell, and Volinsky, 2009). The constant λ is typically learned via cross-validation.

While these methods are designed to predict the rating of a product by a user, determining which users to target with advertisements for a specific product requires predicting the scores for that product by every user, and then choosing the users with the highest predicted score. This can be computationally expensive since as new ratings are input, the nearest neighbors for users or latent factors may change and need to be updated. Also, recommending niche items might be more difficult if the product does not have a high predicted rating for many users or if a product has very few ratings, since there would not be enough information to accurately find neighboring items or incorporate into latent factors.

2.3 Methodology

We propose a different approach, where we take a group of users known to like an item and use their characteristics as a template to find a similar group who will also like the item. This can be done by balancing across a set of covariates using the Balance Optimization Subset Selection (BOSS) framework(Nikolaev et al., 2013).

2.3.1 Balance Optimization Subset Selection (BOSS)

The goal of BOSS is to find a group of units out of a large pool that is similar to a target group. To find the control group, a Mixed Integer Program (MIP) is solved to minimize the difference between the proportion of users in the treatment and control groups having each value of each covariate (Sauppe, Jacobson, and Sewell, 2014). We define a slack variable for each value of each covariate as the difference between the number of units having that value for that covariate in the treatment and control groups. We then seek to minimze all of those slack variables. If the objective value is 0 at optimality, the distributions for each covariate are identical, where perfect balance is found.

Let T be the treatment group, and C be the control pool. The users chosen by the MIP will form the control group, where x_c for all $c \in C$ is 1 if user c is chosen and 0 if not. Let P be the set of covariates, B_i be the set of values for each covariate $i \in P$, $B_{ij} \subseteq T \cup C$ be the set of units having value j for covariate i, and s_{ij} be the slack variables for bin j in covariate i. β is the multiplier for the number of users desired to be in the control group compared to the treatment group. For example, $\beta = 1$ will make the control group the same size as the treatment group. The formulation of the MIP is

$$min \left\{ \sum_{i \in X_b} \sum_{j \in B_i} (s_{ij}^+ + s_{ij}^-) \right\} \tag{11a}$$

subject to

$$\sum_{c \in C} x_c = \beta |T| \tag{11b}$$

$$\sum_{c \in C \cap B_{ij}} x_c + s_{ij}^+ - s_{ij}^- = \beta |T \cap B_{ij}| \text{ for all } i \in P, j \in B_i \tag{11c}$$

$$x_c \in \{0, 1\} \text{ for all } c \in C \tag{11d}$$

$$s_{ij}^+, s_{ij}^- \geq 0 \text{ for all } i \in X_b, j \in B_i \tag{11e}$$

The objective function (11a) minimizes the sum of the slack variables, which minimizes the difference between the covariate distributions of the control group and the treatment group. Constraint (11b) sets the size of the control group, and (11c) sets the slack variables as the difference between the number of users in the treatment group and the control group for each value of each covariate.

The β value is important not only because it affects the size of the control group, but also because it can change the likelihood of finding a group with zero imbalance. As β increases, the number of users in the control group increases. Therefore, the number of combinations of users available to form the same covariate distributions as in the treatment group increase, which can affect the chance of finding perfect

balance. Furthermore, when using the given formulation, β should be an integer so that there are no values with a fractional number of units.

This paper uses BOSS to create a list of users who could be the recipients of targeted advertising for a given movie. To do this, we set the treatment group to consist of users who have liked the movie. The control pool consists of all users who have not seen the movie. From this pool, we wish to choose a control group that consists of users who would also like the movie. The dataset consisted only of movie ratings, so the covariates used were how each user rated different movies. Since the movie ratings reflect the preferences of individuals, finding a control group that has the same opinions on movies as the treatment group should generally result in the control group liking the movie we want to recommend.

Choosing informative covariates is critical to finding a control group who share the preferences of the treatment group. The ideal set of covariates completely separates users who like the target movie from those who do not. For example, if every person who likes the target movie also likes movie B and every person who dislikes the target movie dislikes movie B, movie B would make a good covariate. Conversely, if everybody likes movie B or everybody dislikes movie B, it would make a poor covariate since it does not provide much information about the users who like the target movie. At present, BOSS cannot gracefully account for missing entries, so missing entries are treated as another value when balancing. As such, while not rating a movie can be seen as a choice (this person would prefer to watch other movies), it is less informative than an explicit rating. Missing values in this case are viewed as a choice - we assume that the user chose to not watch that movie, which is an indicator of user preference.

The Netflix dataset was chosen since it was a large publicly available dataset of ratings and because it has been used in prior analysis in the Netflix Prize competition. The dataset reflects real users and their real ratings for movies in a scenario where their ratings actually effected their experience, so false ratings are likely to be less prevalent. The dataset consists of 480,189 users and 17,770 movies with 100,480,507 ratings (forming a 1% completion rate). In the Netflix dataset, the covariates in use are the ratings for different movies.

If three covariates are used, an example group consisting of 1000 users might be characterized as follows (with each rating being one of 1, 2, 3, 4, or 5):

Movie 1	# rated
1	100
2	100
3	300
4	400
5	100

Movie 2	# rated
1	200
2	200
3	200
4	200
5	200

Movie 3	# rated
1	400
2	200
3	200
4	100
5	100

The methods by which movies were chosen as covariates are given in section 2.3.2.

To measure the results of BOSS, we chose the treatment group from the users who rated the target movie 5 out of 5 using uniform random sampling without replacement. The control pool consisted of every user who had rated the target movie, excluding the treatment group. By only including users whose rating of the target movie is known, the ratings of the users in the control group chosen by BOSS would be observable, and hence could be used to determine if BOSS was providing any improvement over random selection. $\beta = 1$ was used in order to keep the size of the control group small.

One reason to use BOSS is that it can provide results with only a small number of covariates. By only requiring a few attributes, BOSS is able to scale to large datasets without requiring excessive computational time. Another advantage is that BOSS can handle users with very little data or history, since users in the treatment group will often also have a small history due to being new users or users who have not provided much data. Since BOSS operates on the group level, the control group provided by BOSS often incorporates users with limited history, which means that it does not necessarily suffer from the same cold start problem that collaborative filtering suffers from, while also not requiring the construction of profiles like content filtering. While the results will often be better with the inclusion of more user data, the ability to gain some useful information even with limited user data is an advantage when compared to other systems. However, these advantages do come at the price of individual detail, since BOSS selects users by selecting a group as a whole and not each user independently Therefore, the method is suitable for choosing a group, but not for displaying a predicted rating to an individual user for an item, as Netflix does. Furthermore, not every individual in the group selected by BOSS will like the target movie - as a group they may like it, but there will likely be certain invididuals who do not.

2.3.2 Covariate Selection

BOSS was applied using two different methods of covariate selection to examine the impact that covariate choice has on the units selected by BOSS. The first method was designed to minimize the number of missing ratings in the control pool. To do this, the covariates chosen were the movies for which users in the treatment

group had the most ratings. These movies were found by taking all the ratings by every user in the treatment group and selecting the most-rated movies.

The second method was to find movies similar to the target movies to use as covariates. The intuition is that how users rated similar movies would be a strong reflection on how they would rate the target movie. In order to find similar movies, the first step was to create a set of features that could be used to describe each movie. Rather than create these features by hand, we opted to use the method given by Simon Funk(Funk, 2006), which uses matrix factorization to decompose the entire matrix of user-movie ratings into a matrix of user features and a set of movie features, where the product of a user's features and a movie's features would be the rating of that movie by that user. In order to not contaminate the results, 10,000 ratings for each of the 2,000 most-rated movies were separated into a training set which was used to compute the matrix factorization. The ratings not included in the training set were used to form a test set which BOSS was run on. The training set ratings were then not used when running BOSS. As suggested by Funk (2006), 40 features were learned for both users and movies. In order to compute the factorization, a user and movie matrix were created where the number of rows was the number of unique users or movies in the training set and the number of columns was the number of features (40). Let U be the user matrix and M be the movie matrix, where U_{ij} is the value of feature j for user i and M_{ij} is the value of feature j for movie i. Let U_i be the ith row of U and M_i be the ith row of M. The ratings in the training set are then iterated over, with each rating r of movie m by user u updating feature i using the update rule

$$err = r - U_u \dot{M}_m \tag{12}$$

$$U_{ui} = U_{ui} + \alpha(err * M_{mi} - K * U_{ui}) \tag{13}$$

$$M_{mi} = M_{mi} + \alpha(err * U_{ui} - K * M_{mi}) \tag{14}$$

where α is the learning rate set to 0.001 and K is the regularization parameter set to 0.015. Each feature was trained by iterating over the training set 100 times.

The resulting movie matrix M contains the value of 40 features for each of the 2000 movies in the training set. These features were then used to find similar movies by using the k-Nearest Neighbors method. Two different similarity metrics were tested - the Euclidean distance and the Cosine Similarity. The k movies chosen as covariates were the k nearest neighbors of the target movie. BOSS was then run using each of the three different sets of covariates (most-rated, Euclidean kNN, Cosine Similarity kNN) where the treatment group was composed of 5-out-of 5 ratings from the training set and the control pool consisted of ratings in the test set. The size of the treatment group was set to 10% of the size of the control pool, and β was set

to 1, so the control group selected would be 10% of the control pool.

2.4 Results

Several experiments were performed to determine the number of covariates to use, which covariates to use, and the performance of BOSS. The first variable to consider was the number of covariates that should be used with BOSS. To do this, BOSS was run on the same target movie using the same treatment group with 1, 5, and 10 covariates, and the results using the different numbers of covariates were compared. The covariates used were the movies most rated by users in the treatment group. For example, if one covariate was used, the covariate would be the movie with the most ratings by users in the treatment group. The 10 covariates case included the 10 movies with the most ratings by users in the treatment group. The ratings of the chosen group (control group) were then compared to the ratings of the users in the control pool to determine if any improvement was made over random selection. The ideal result would be for everybody in the control group to rate the movie 4 or 5, since in that case everybody chosen would have liked the movie. The result for the most viewed movie is depicted in Figure 11. The left graph shows the percentage of users who gave the target movie each rating. The control group is the bar on the left, and each of the next 3 bars are with BOSS using 1, 5, and 10 covariates respectively.

Figure 11 shows that as the number of covariates increases, the rating of the target movie by the users in the control group shifts to the right, meaning that adding more covariates improves the average rating of the control group chosen by BOSS. This result is intuitive, since adding more covariates creates a more thorough description of users in the treatment group, and hence should select users for the control group that are more similar than randomly selecting users.

In order to assess the improvement BOSS provides over random selection, BOSS was run on each of the 2000 movies used to construct the training set using each of the described covariate selection methods. Since there were many optimal solutions to BOSS, 10 optimal solutions were collected for each movie. Movies with fewer ratings were excluded because measuring the success of the program required only including users who had already rated the movie so that their rating could be used to compare to random selection. However, this limitation would not necessarily be present in an actual application of BOSS where the control pool would consist of all users who had not rated the movie. All users with ratings for the target movie were included in the problem regardless of their ratings for other movies (or lack thereof).

To determine the success of BOSS, the mean rating of the target movie by users in the control group and the control pool were calculated for each of the 2000 movies. The difference in means is indicative of the overall improvement (or lack thereof) that BOSS provided. The difference in means between the control

group and the control pool is depicted in Figures 1, 2, and 3, which show the distribution of the difference in mean across all 2000 movies in the training data.

As depicted in Figures 1, 2, and 3, the control group chosen by BOSS has a higher average rating than the control pool in general. The higher mean rating shows that using BOSS to select a subset of users to target provides a higher expected rating for the target movie than using a random selection from the control pool. Table 1 gives the mean and standard deviation of the improvement generated by BOSS. The table shows that the kNN methods performed roughly the same as each other, and both performed significantly better than the Most-Rated covariate selection method. Combining this with the earlier results from Figure 11, we can see that this is caused by a noticeable reduction in the selection of users who rated the movie one or two. This is important, since users who rated the movie one or two are the ones most likely to be disgruntled with receiving a recommendation for the movie.

One observation is that running BOSS almost never gave a group with a lower mean rating than the pool, meaning that using BOSS was advantageous in nearly every case. This is an important observation, since an overall improvement in mean would not necessarily be advantageous if it frequently gave a negative result.

To assess the significance of the results, a 1-sample t-test was used to compare the average ratings of the control group and the control pool for each movie using the mean of the control pool as the population mean. The null hypothesis for the test is that the control group was randomly drawn from the control pool. Figures 4, 5, and 2.6 show the distribution of t-values for each of the covariate methods. In all three cases, the t-values are predominantly very large, leading to the conclusion that the improvement by BOSS over random selection is statistically significant.

2.4.1 Parameter Experimentation

In order to find the parameters for BOSS that provided the most improvement over random selection, runs were conducted with varying treatment group sizes and different β values. Values of β of 1, 5, 10, and 20 were tested, and it was determined that changing β within that range did not affect the result by observing that the mean of the control group was not improved by increasing or decreasing the β value. The size of the treatment group was varied by changing the fraction of users who liked the movie that were selected for the treatment group. Percentages of 10%, 25%, and 50% were tried; the 10% setting had the best results when the means of the control group for each percentage were compared. When the percentage was raised, the mean of the control group converged to the mean of the control pool, which meant that BOSS was performing no better than random selection. One hypothesis as to why this would be is that when the treatment group consisted of a high percentage of users who liked the movie, there were not enough

users left in the control pool who liked the movie, and therefore the mean rating of the control group was lowered. In all experiments, only results where perfect balance was achieved (via an objective value of 0) were considered, as the implications of imperfect balance are not clear. When using the most-popular covariate selection method, imperfect balance was found in ¡1% of cases, while it occurred in 10% of cases when using the kNN covariate selection methods.

2.4.2 Standardization of Data

Typically, the ratings by a user do not reflect their rating on an absolute scale. There are biases introduced - some users tend to rate higher than others, while some products also tend to be rated higher than others. Since the goal is not to predict the user rating, but to predict if a user would like a movie, it is important to separate a user's rating habits from their preferences. A user that systematically rates higher than others should have their ratings lowered to conform to other users so that comparisons can be made correctly. Likewise, movies that are rated highly by everybody should have their ratings systematically lowered so that the average rating of a highly rated movie is the same as the average rating of a less-liked movie to avoid recommending purely popular movies and recommend those that are specifically in line with a particular user. In order to correct these systemic trends, some rating standardization was done to better compare users. The first standardization method attempted was based on the assumption that the ratings should conform to a normal distribution, where each rating for movie i by user u r_{iu} was changed to a standardized rating r'_{iu}:

$$r'_{iu} = \frac{r_{iu} - \mu}{\sigma} \tag{15}$$

where μ is the mean of all the ratings for movie i and σ is the standard deviation of all ratings for movie i. However, this model did not represent the data well, since movies rarely were symmetrically rated - they tended to have a skewed rating distribution by having either more positive or more negative ratings - and did not affect the result in a positive manner. The results of this standardization technique produced a control group that often had a mean rating lower than the mean rating of the control pool, suggesting that the technique did worse than random selection. An effective standardization was found by applying the standardization methods outlined by Bell, Koren, and Volinsky, 2007 for their winning entry in the Netflix Prize competition. Each rating was broken down into a sum of observable effects and a residual, where the residual is the portion of the rating not explained by any of the calculated effects. This residual is the part we are interested in using, as it represents the portion of the rating unique to this user-movie combination. In our standardization process, we separated each rating into a sum of a user effect, a movie effect, and a residual. Let r_μ refer to the global mean for all ratings in the dataset, M to the set of movies, U to the set

of users, and r_{iu} be the original rating of movie $i \in M$ by user $u \in U$. The movie effect γ_i for movie $i \in M$ centers the mean rating for each movie by subtracting r_μ from the movie's mean. An example of this effect is rating a movie 4 where the mean rating is a 4.5 which suggests that this user likes the movie less than the average user, despite the relatively high rating. The user effect η_u for user $u \in U$ removes the user bias by subtracting the global rating mean from each user's rating mean. For example, if a user gives a mean rating of 2, their rare usage of 4 is much more impactful than a 4 from a user with a mean rating of 4. The user effect is particularly important to model because it allows the users to be more directly compared. If one user systematically rates higher than another user but has the same preferences, an unaccounted-for user effect would lead to the incorrect conclusion that the two users have difference preferences. The user effect was calculated after the removal of the movie effect from the data. Each normalized rating ϕ_{iu} of movie i by user u is given by

$$\phi_{iu} = r_{iu} - r_\mu - \gamma_i - \eta_u \text{ for all } i \in M, u \in U \tag{16}$$

What is left after the removal of these two effects is a rating for each movie that allows more accurate comparisons to be made between ratings.

A result of standardization is that the ratings are no longer one of five discrete values like the raw data, but instead occupy a continuous range of values that varies from movie to movie. This made the data unsuitable for the previously used formulation of BOSS, because that formulation is designed to match counts of covariate values. Therefore, the new ratings were clipped to the interval [-2, 2] and then rounded to the nearest integer. This was done to enable perfect balance to be found in the majority of cases - a more granular rounding allows for higher distinction at the cost of perfect balance. BOSS was then solved on this standardized data using the Most-Rated and kNN-Euclidean covariate selection methods. Again, 10 optimal solutions were kept for each of the 2000 solved movies. Figures 7 and 8 show the difference in means and figures 9 and 10 show the t-tests for the standardized data. Once again, we can see that the kNN covariate selection method performed significantly better than the most-rated method.

2.4.3 Computational Complexity

Sauppe, Jacobson, and Sewell (2014) proves that BOSS is NP-hard. The problem size is determined by the number of covariates, unique covariate values, and number of units in the control pool. In the first presented formulation, there are $|C|$ binary variables representing user selection, where $|C|$ is the number of users in the control pool, as well as two slack variables for each value of each covariate. The standard BOSS formulation has one constraint for each value of each covariate, as well as one constraint to set the size of the control group. Therefore, the MIP model for BOSS scales with the number of users in the control pool as well as

the number of values of each covariate being used (or the number of bins if the covariate values are binned). Since in the Netflix dataset all the ratings are between one and five, the total number of variables in the BOSS formulation is $10|X|+|C|$ where X is the set of covariates and C is the set of users in the control pool. The total number of constraints is $5|X|+1$ (not including the binary constraints). To establish an estimate for the runtime of BOSS on a given movie, movies with 232,944, 50,196, and 25,857 ratings were run 10 times each and the runtimes recorded in Table 3. These movies were chosen to provide samples across a spectrum of sizes to determine how the runtime is affected. All runs were performed with a computer using an Intel Xeon E3-1246 quad-core processor at 3.5GHz using the GUROBI solver to solve the MIP.

From table 3 we can observe that the runtime of BOSS did in fact scale with the number of included units. However, in these samples we can see that the growth in runtime was not very far over linear, since between the smallest and largest problem a 10x growth in included units resulted in a 10.7x growth in mean runtime. This shows that this MIP formulation scaled reasonably well with problem size, although there is variability in the runtime, where the max runtime is almost twice the min runtime for each sample. This shows that the specific units in the treatment group and control pool can impact the runtime significantly.

2.5 Conclusions and Further Research

By using the Netflix dataset, BOSS demonstrates its ability to select a subset of users for targeted advertising for a product. BOSS was run both using the raw data over a sample of 2000 movies and on standardized data over a sample of 2000 movies in order to verify its applicability. The control groups chosen by the runs were then compared to random selection to see if BOSS was an improvement. Finally, the runtimes of BOSS were observed to examine its scalability as much as could be determined using this single dataset.

Running BOSS on the raw ratings data was shown to provide a statistically significant improvement over random selection, while running in reasonable time on even the movie with the most ratings in the dataset. The longest run observed was 18.53 seconds, while the shortest was 0.77 seconds. In order to account for the differences in user and movie ratings amongst individual users and movies, a standardization scheme was used to remove bias in the ratings themselves. Solving BOSS on this standardized data also led to a statistically significant improvement.

The main advantage to BOSS over other solutions is that BOSS does not require any model training beyond the original learning of movie features when using the kNN covariate selection methods. While the matrix factorization itself can be used to predict ratings, it does require the model to constantly be trained as new users, movies, and ratings enter the system. BOSS, meanwhile, can be run on the raw rating values. This is a huge advantage when the dataset gets large, as doing the matrix factorization on a large dataset is

computationally expensive.

One limitation of BOSS is its lack of ability to predict the rating for a single user. This group methodology enables the creation of a target group rather than determining information about individuals. The second limitation is the way BOSS handles missing data, since BOSS treats missing data as its own value. The argument could be made that the lack of a rating provides information, since the user either chose not to watch the movie or chose to watch and not rate it, but that may not be acceptable in all cases. This may be alleviated in cases where more data is available, such as whether a user chose to watch a movie and not rate it, saw the movie listing but chose to not watch it, or never even saw the movie listing as an option.

The largest area for potential improvement over these results is the choice of covariates. For this experiment, the covariates used were the movies most seen by users in the treatment group and movies calculated as similar to the target movie. However, other options should be tested, such as also including movies that are very different than the target movie. Also, one note on the technique is that in the case of missing data, the lack of a rating was taken as a choice by the user to have not watched the movie. Therefore, it is implicitly included in the balancing. If this assumption is deemed invalid in a particular use case, its effects could be mitigated by having more data (e.g. whether a user has watched a movie and decided not to rate it) or by limiting covariates to movies where the rating is known.

The last point to be touched on is data quality - more personal information on the users may lead to larger improvements over random selection, since the Netflix Prize dataset is devoid of any demographic information. Traits such as age, profession, and location might provide a superior selection mechanism for the control group, since it would allow for more accurate profiling of the users in the dataset.

2.6 Figures and Tables

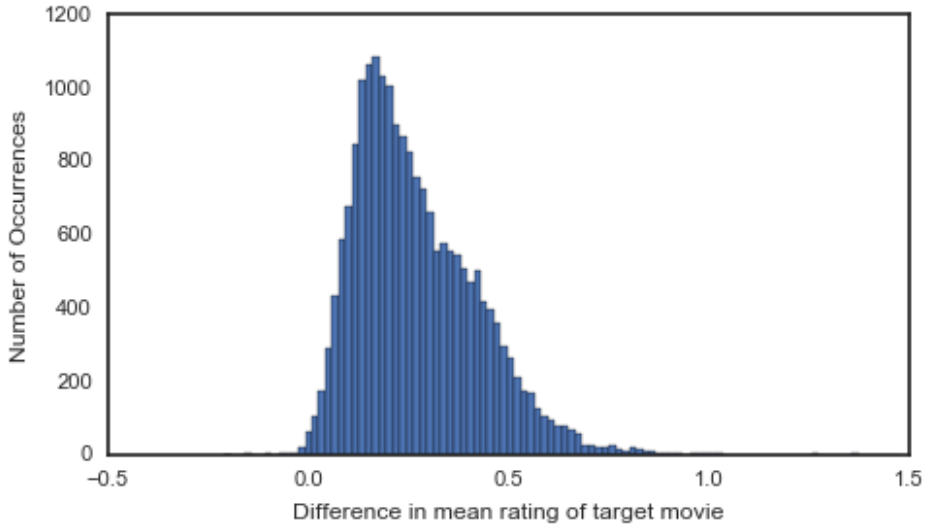

Figure 1: Difference in Mean between Control Group and Control Pool (Most-Rated)

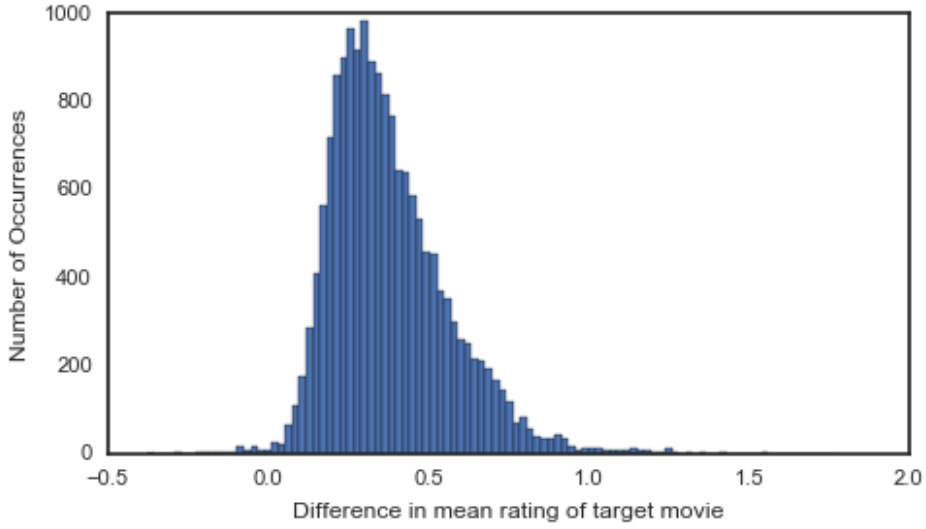

Figure 2: Difference in Mean between Control Group and Control Pool (kNN - Euclidean)

Method	Mean(Difference)	Stdev (Difference)
Most-Rated	0.271	0.154
kNN (Euclidean)	0.384	0.198
kNN (Cosine)	0.384	0.196

Table 1: Improvement by BOSS over random selection

Method	Mean(Difference)	Stdev (Difference)
Most-Rated	0.133	0.132
kNN (Euclidean)	0.253	0.150

Table 2: Improvement by BOSS over random selection using standardized data

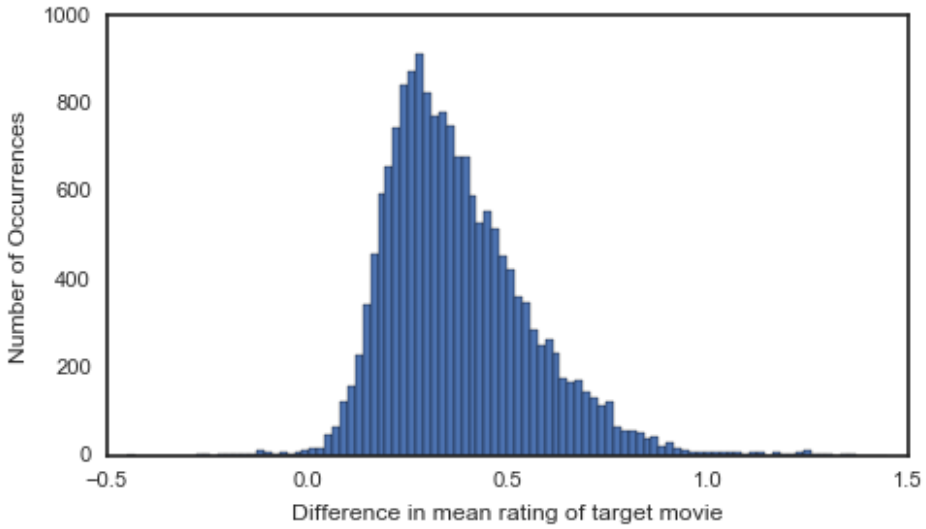

Figure 3: Difference in Mean between Control Group and Control Pool (kNN - Cosine)

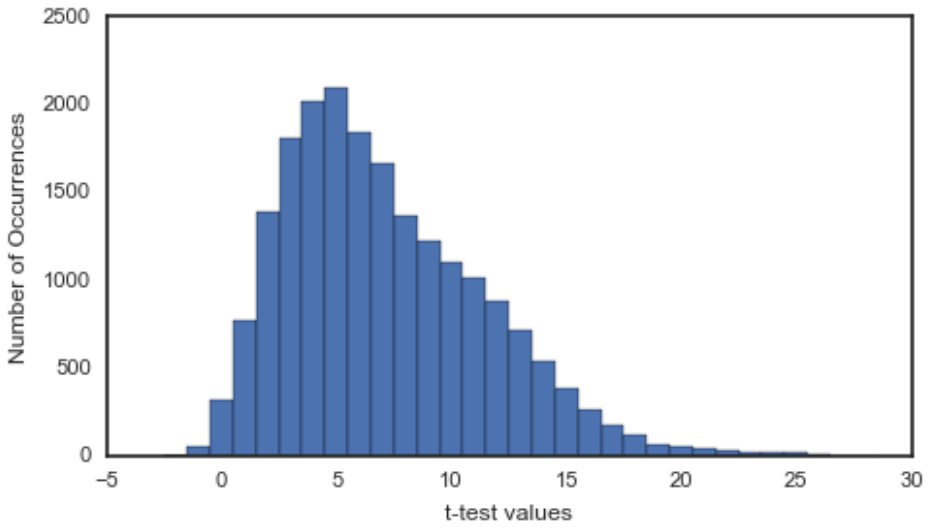

Figure 4: T-Test between Control Group and Control Pool (Most-Rated)

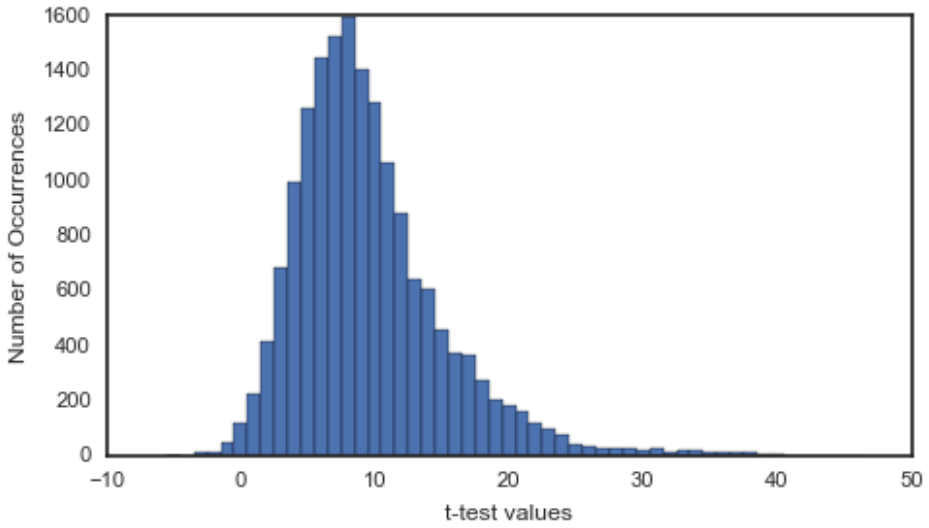

Figure 5: T-Test between Control Group and Control Pool (kNN - Euclidean)

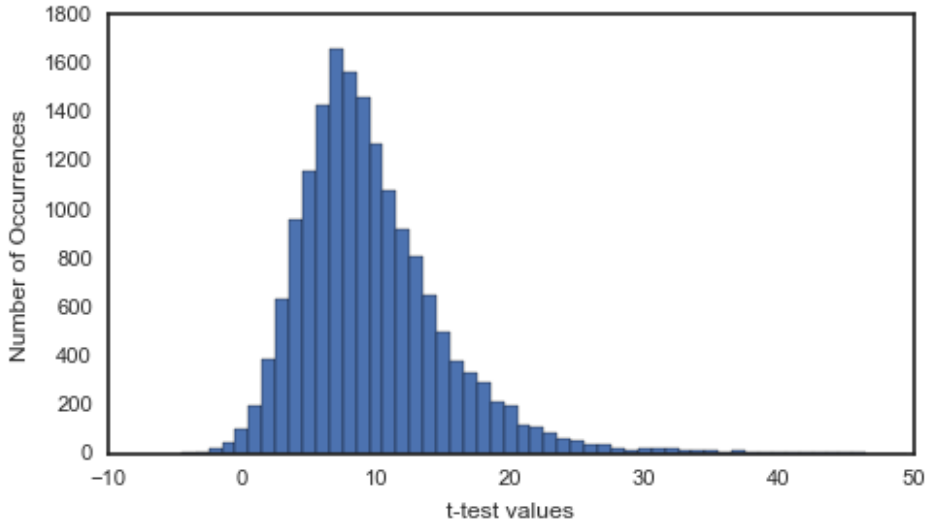

Figure 6: T-Test between Control Group and Control Pool (kNN - Cosine)

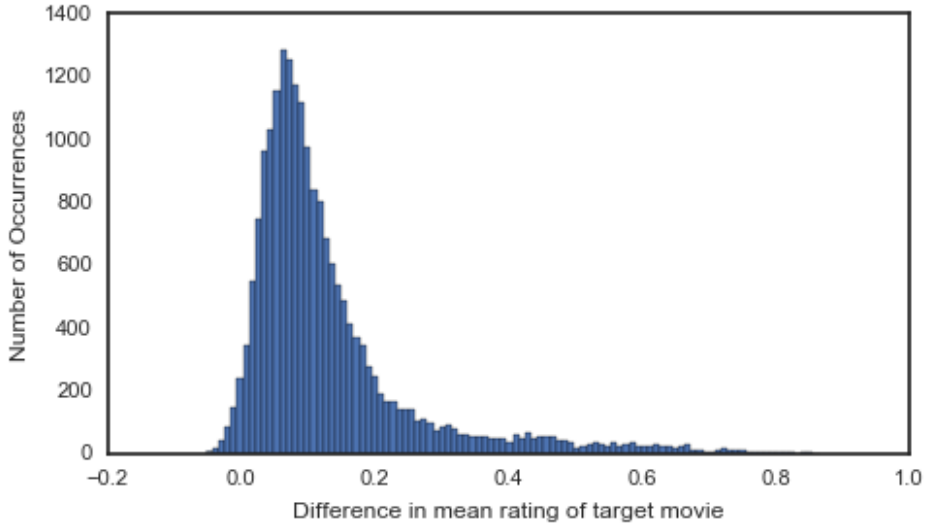

Figure 7: Difference in Mean between Control Group and Control Pool (Most-Rated)

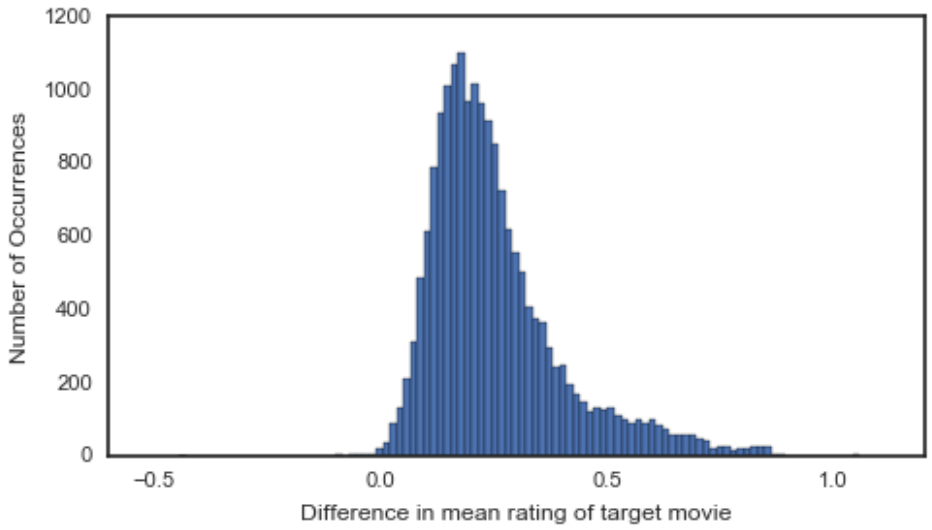

Figure 8: Difference in Mean between Control Group and Control Pool (kNN - Euclidean)

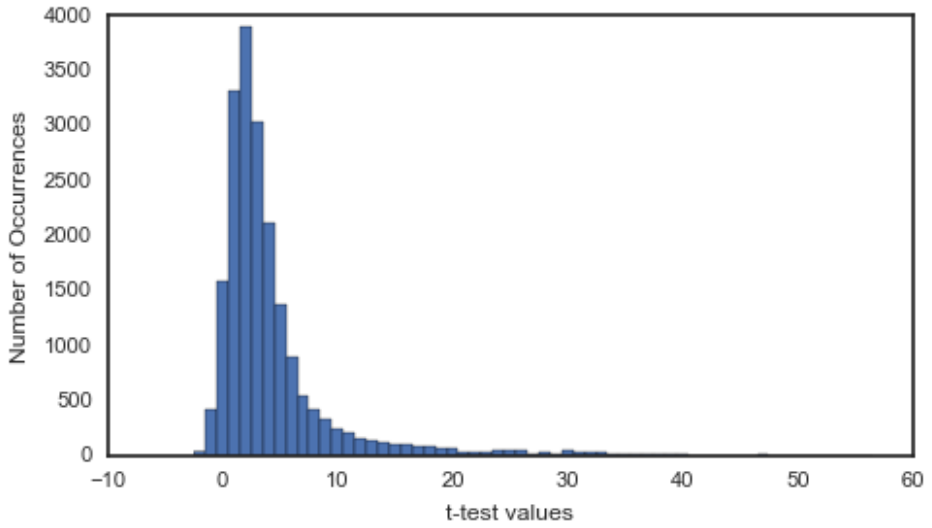

Figure 9: T-Test between Control Group and Control Pool (Most-Rated)

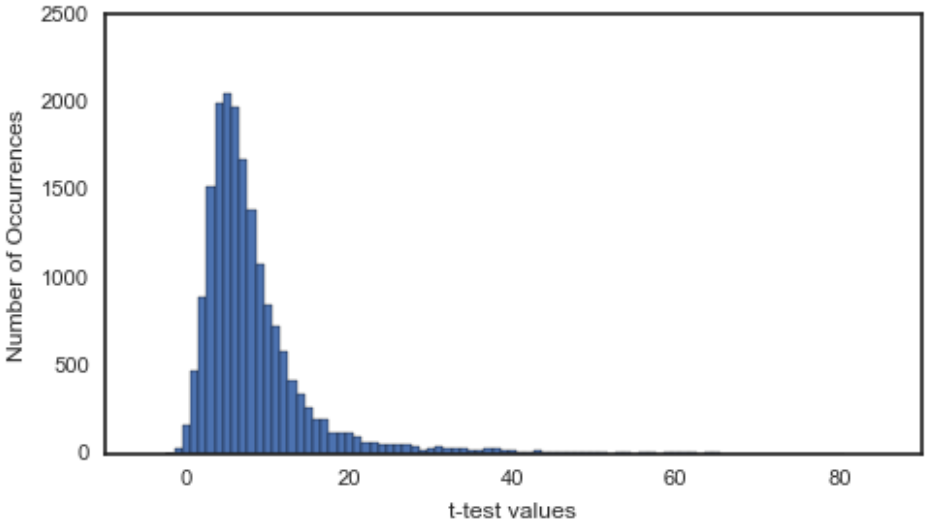

Figure 10: T-Test between Control Group and Control Pool (kNN - Euclidean)

# Ratings	**Min** (*sec*)	**Mean** (*sec*)	**Max** (*sec*)
232,944	10.24	12.51	18.53
50,196	1.90	2.56	3.53
25,857	0.77	1.17	1.45

Table 3: BOSS Runtimes

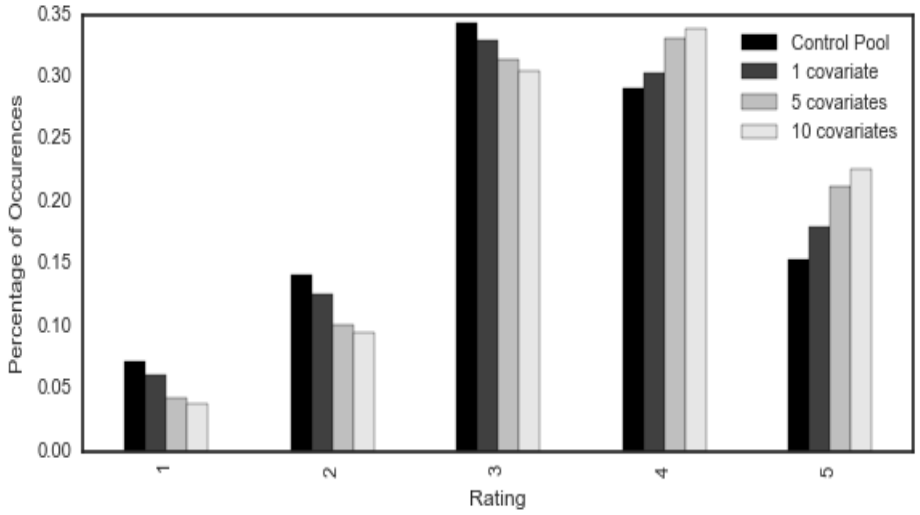

Figure 11: Effect of Covariate Choice

3 Identifying NCAA Tournament Upsets

3.1 Introduction

The men's college basketball championship tournament, henceforth referred to as *the tournament*, is held annually by the National Collegiate Athletic Association (NCAA). The tournament attracts a tremendous amount of attention nationally from the public and the media, which has resulted in the event being commonly known as *March Madness*. People all over the country engage in the tournament both by supporting their favorite team and by attempting to predict the outcome. Sites such as ESPN (ESPN, 2015) and Yahoo (Yahoo, 2015) host bracket competitions, where people submit their predictions for the outcome of each game in the tournament. In 2015 alone, people submitted approximately 70 million brackets to various sites (Wartenberg, 2015). Accurately predicting the outcome of games can also be financially lucrative, with Americans wagering approximately $9 billion in 2015 (Marino, 2015). In 2014, Quicken Loans partnered with Yahoo to offer $1 billion to anybody who could create a bracket with every game predicted correctly (Yahoo, 2014). Both the pride from being correct and financial opportunities have incentivized many individuals and companies to develop models to predict the outcomes of the tournament games.

The world of sports forecasting can be a daunting one for those people not familiar with the sport and current teams. Newcomers attempting to learn about the teams in the tournament are faced with a copious amount of statistics, team rankings, and expert opinions. To help users create a bracket, news and sports sites such as ESPN and Fivethirtyeight.com (FiveThirtyEight, 2015) make their own predictions publicly on how the tournament will proceed. However, while they do disclose some components and relationships that go into their predictive models, a large portion of the models are proprietary. Even the revealed portions of the models involve a multitude of factors that are prohibitive for a newcomer to obtain and use. The use of a few key statistics that are easy to acquire and understand would allow both experts and novices to make forecasts based on the same data.

This paper proposes a technique to select potential upsets using only a small number of publicly available statistics by identifying match-ups in the current year that exhibit characteristics similar to those exhibited by historical round-of-64 upsets. The differences in season statistics between the two teams in each historical upset are used to build a profile of past upsets, which is then compared to first round games in the current year to find match-ups that are most similar to historical upsets. By limiting the potential characteristics to game statistics, the technique can be back-tested to ascertain its accuracy. Testing was done by generating predictions for each year from 2003 to 2015 using only data that would have been available at the time for each year. This technique is shown to outperform the random selection of upsets, and the results obtained are reproducible using freely available information from TeamRankings.com (TeamRankings, 2015). By

examining historical upsets, our technique is able to identify characteristics that allow a weaker team to beat a stronger team, and then find games in a given year's tournament that exhibit those characteristics. Taking the specific matchup in each game into account allows us to identify upsets with greater accuracy than weighted random selection would allow.

This paper is organized as follows. Section 3.2 describes current techniques for predicting the outcome of games. Section 3.3 describes the method by which Balance Optimization Subset Selection is used to choose potential upsets. Section 3.4 summarizes the results of the proposed technique by providing the predictions made by the technique. Section 3.5 provides concluding remarks.

3.2 Background

3.2.1 Rating Systems

Several team rating systems that quantify the skill of teams have been introduced and popularized, including ESPN's *Basketball Power Index (BPI)* (Oliver, 2013), Ken Pomeroy's *pythagorean ratings* (Pomeroy, 2012), and Jeff Sagarin's *predictor ratings* (Sagarin, 2015). These rating systems focus on assigning a numeric rating to each team that estimates how successful that team will be in games. The premise behind these systems is that a team with a higher rating is strong than a team with a lower rating, where the difference between the rating of the two teams indicates the difference between the strength of the teams.

The Basketball Power Index (BPI) was introduced in 2013 by ESPN and touted as "a little more refined than any other existing power ranking."(Oliver, 2013) While the exact formula for calculating a team's BPI is not reported in the literature, ESPN does reveal some of the components of the ranking. The BPI includes such information as a team missing an important player during a game, whether the game is home or away, whether the game was a blowout or a close game (Oliver, 2013), the pace of the game, and the strength of a team's schedule (how strong a team's opponents were) (Volner, 2013).

Ken Pomeroy, owner and operator of kenpom.com, scores teams based on a *pythagorean winning percentage* (Pomeroy, 2012), which is the expected fraction of games a team should win against an average team. To calculate this percentage, he uses the *adjusted offensive efficiency (AdjO)* and *adjusted defensive efficiency (AdjD)* of a team. The adjusted offensive efficiency is an estimate of the number of points a particular team would score per 100 possessions against an average team (as assessed by Pomeroy). The adjusted defensive efficiency is an estimate of the points allowed per 100 possessions by a team against an average team. The method of computing these adjusted values is not reported in the literature. The adjusted offensive effiency and adjusted defensive efficiency are then combined into the pythagorean winning percentage using

the formula

$$\text{pyth} = \frac{AdjO^{10.25}}{AdjO^{10.25} + AdjD^{10.25}}. \tag{17}$$

These ratings systems are used to determine the outcome of games. In a match-up between two teams, the team with the higher rating is predicted to have a higher likelihood of winning a game. The magnitude of the difference in the ratings is also used to determine how likely each team is to win. For example, Fivethirtyeight.com combines seven different ratings to predict the likelihood of one team beating another (Silver, 2015).

3.2.2 Match-up Analysis

An alternative to predicting game outcomes by comparing the rating of two teams is to compare the two teams in a match-up directly.

The tournament is divided into four regions, and each team in the tournament is given a *seed*, which is an estimate of the rank of a team in their respective region of the bracket determined by the selection committee prior to the tournament. The team deemed by the committee to be the strongest in each region is given a seed of 1 and the team deemed to be the weakest in each region is given a seed of 16. We define a team with a small numeric seed as having a *high seed* and teams with a large numeric seed as having a *low seed*. Therefore, a team with seed one is the highest seeded team in its region and a team with seed 16 is the lowest seeded team in its region. The games of interest to this paper are upsets, which are games in which a low seeded team beats a high seeded team. Upsets are defined by ESPN as games where the difference between the seed of the winning team and the seed of the losing team is at least five (Keating, 2013). ESPN looks for potential upsets by looking for teams that are stronger than their seed would suggest or by finding match-ups where the weaker seeded team has a strength that could exploit the weakness of a stronger team (Brenner and Keating, 2015). ESPN defines four categories of high seed teams that are capable of losing (Brenner and Keating, 2015):

- Power Giants: Strong offensive rebounding, average defensive rebounding, do not force many turnovers
- Gambling Giants: Strong offensive rebounding, weak defensive rebounding, force many turnovers
- Pack-line Giants: Average offensive rebounding, do not force many turnovers, good defensive rebounding
- Generic Giants: Generally skilled, not specifically strong in offensive rebounding or generating turnovers

ESPN also defines four categories of low seed teams that have the potential to upset (Brenner and Keating, 2015):

- Generic Killers: Decent teams with no especially strong rebounding, turnovers, shooting, or defense
- Slow Killers: Strong offensive rebounding, limit opponent shooting, neither generate steals nor shoot 3-point shots
- Perimeter Killers: Strong 3-point shooters, generate lots of steals, weak offensive rebounding, weak at limiting opponent shooting
- High-Possession Killers: Limit opponent shot accuracy, strong offensive rebounding, do not shoot many 3-point shots

These categories are then analyzed to see which *Giants* (high seed teams) could fall to which *Giant Killers* (low seed teams). ESPN's conclusions are shown in Figure 12, where an arrow from a Giant to a Giant Killer means that the Giant is weak against that Giant Killer and an arrow from a Giant Killer to a Giant means that the Giant Killer is strong against that Giant.

ESPN does not elaborate on exactly how each team is placed into any of the above categories.

There has also been research into quantitatively predicting the probability of one team winning against another. Kaggle (Kaggle, 2015), a website that hosts data science competitions, ran a college basketball tournament prediction contest in 2014 and 2015. In Kaggle's competition, participants were asked to compute the probability of each team beating each other team in the tournament, but were only scored on those matches that actually occurred. Since there are 68 teams in the tournament (including the play-in matches), participants made probabilistic predictions for each of the 2278 potential team pairings that could occur. Each match was weighted equally, unlike traditional bracket scoring where predicting the winner of the tournament is worth several times as many points as predicting the outcome of a round-of-64 match. The advantage of Kaggle's system is that it made predicting round-of-64 upsets correctly more advantageous than a traditional bracket where the later rounds are much more important due to later rounds typically being worth more points. The Kaggle competition winners (Lopez, 2015) used a logistic regression model incorporating the Las Vegas point spread (the expected margin of each game) given by Covers (Covers, 2015) and Ken Pomeroy's efficiency ratings.

The downside to the existing methodologies for predicting the outcome of a game is that they are difficult to replicate due to their opaque nature. The method for calculating the Las Vegas point spread is not publicly available, and neither is the exact formula for ESPN's BPI. Also, while using factors such as strength of schedule seems useful, it introduces its own biases since it is not an objective statistic, but rather relies on people to determine the relative strengths of teams who have never played against each other. ESPN outlines qualities of Giants that may fall and the Giant Killers that may upset them, but does not elaborate on the

methods used to identify these teams. This paper aims to provide an approach to predicting upsets that uses only statistics that can be obtained from watching the games, and also self-identifies the important factors in predicting upsets without user intervention.

3.3 Methodology

In this paper, we define an upset as a team seeded 13, 14, or 15 winning a game in the round-of-64 (compared to ESPN's 11 or higher). A *k-seed game* refers to a game in which one of the teams has seed k. We exclude 16-seeded teams because no 16-seed has ever won a round-of-64 game. This allows the focus to be placed on identifying characteristics of a much smaller number of upsets than if 11 and 12 seeded teams were included. Between 1985 and 2015, 52 of the 372 games played involving 13-15 seeds have been upsets. This is an average of 1.67 upsets per year. Therefore, we aimed to identify two match-ups each year that are similar to past historical upsets. All raw data was obtained from TeamRankings.com (TeamRankings, 2015) and are obtained as of Selection Sunday for each year (so no tournament game statistics are included in the data). The proposed technique can be described as a series of four steps.

1. Computing match-up statistics: Compare the two teams playing in each game using team statistics.
2. Identifying useful match-up statistics: Use an Extra-Trees Classifier to select a set of match-up statistics $\mathcal{S}$ that are strong indicators of historical performance.
3. Finding similar match-ups: For each subset of match-up statistics $S \subseteq \mathcal{S}$ with $|S| = 4$, use Balance Optimization Subset Selection to identify three match-ups that are similar to historical upsets on the match-up statistics in S.
4. Combining models: Identify combinations of match-up statistics $S \in \binom{\mathcal{S}}{4}$ that performed well historically and use them to select two match-ups as potential upsets.

The following subsections will elaborate on each of the four listed steps.

3.3.1 Computing Match-up Statistics

Since play styles in college basketball can change from year to year, we use ordinal rankings instead of the raw statistic value when looking at team statistics. For example, instead of using 58.4% as Notre Dame's Two Point Percentage in 2015, we use the fact that it was the highest Two Point Percentage of any team in 2015, and assign it a value of 1. Gonzaga, having the 2nd highest Two Point Percentage in 2015, would be assigned a value of 2 for that statistic. Relative ranking allows teams from different years to be compared

while accounting for the way that the league as a whole may change. Relative rankings also have the advantage of forcing the range of values for each statistic to be the same (1 to the number of teams barring ties). In the event that multiple teams have the same value for a statistic, they are assigned a rank that is the average of the ranks of that value. For example, if three teams shared the third highest value for a statistic, the three teams would all be assigned $(3 + 4 + 5)/3 = 4$ for that statistic. The team with the next highest value for that statistic would be assigned a rank of 6.

Rather than looking at a team's statistics, it is more useful to look at how a team's statistics compare to those of its opponent. One team averaging a very high number of steals may signal a high number of scoring opportunities, but if the opponent team averages a similar number of steals, neither team is likely to gaun an advantage over each other by relying solely on steals. To find the differences between the teams, we subtract the ordinal ranking of each statistic for the higher-seeded team from those of the lower-seeded team for each game. Comparing the teams in the match-up reveals gaps in the statistics of the teams, such as if one team shoots many more three point shots or one team forces many more turnovers than the other. These match-up statistics, rather than team statistics, are then used to find potential upsets. By observing which statistics have gaps that lead to upsets in historical games, games in the future can be identified as having the potential for being an upset.

3.3.2 Identifying Useful Match-up Statistics

TeamRankings.com (TeamRankings, 2015) tracks 115 different statistics for each team going back to the 1997-1998 season. The first step is to find a small set of statistics that are correlated with upsets. Trying every combination of statistics is infeasible, as there are $2^{115} - 1$ possible combinations. To find a subset of statistics that are useful for selecting potential upsets, an extra-trees classifier (Geurts, Ernst, and Wehenkel, 2006) was trained on round of 64 tournament matches from 1998 to 2015, where the classifier was built using all 115 match-up statistics. An extra-trees classifier builds a large number of decision trees that, in our case, use the various match-up statistics to differentiate upsets from non-upsets. The extra-trees classifier was chosen for its resistance to overfitting (Geurts, Ernst, and Wehenkel, 2006) and because it allows us to measure the importance of each feature for the purposes of classification. The resistance to overfitting is due to the nature of the classifier, which trains each tree on a random subset of the data using a random subset of input features. Since each training example and feature is excluded in many of the trees, the classifier avoids learning to overfit the training set provided a sufficiently large number of trees.

The classifier was built using 100,000 decision trees with $\sqrt{115}$ features sampled for each split and two samples required to split each node (as suggested in Geurts, Ernst, and Wehenkel (2006)). The importance of each feature can then be extracted from the classifier using *Gini Importance* (Breiman et al., 1984). The

implementation used for both the classifier and feature importance was the extra-trees classifier in Python's scikit-learn library. Based on preliminary experiments testing different numbers of match-up statistics with subsequent steps of the technique, we opted to use the resulting 15 most important features for identifying upsets. These features were (in descending order of Gini Importance)

1. Effective Possession Ratio
2. Games Played
3. Extra Scoring Chances per Game
4. Opponent Floor Percentage
5. Personal Fouls per Possession
6. Opponent Steals per Defensive Play
7. Assist / Turnover Ratio
8. Personal Fouls per Defensive Play
9. Opponent Steals per Possession
10. Opponent Average Scoring Margin
11. Average Scoring Margin
12. Opponent Three Point Percentage
13. Steals per Defensive Play
14. Steals per Possession
15. Average 2nd Half Margin

These statistics are defined in the appendix. Let $\mathcal{S}$ be the set of these match-up statistics. It should be noted that the extra-trees classifier was trained using games from 1998-2015 rather than training the classifier to identify features for each year separately. The decision to train the extra-trees classifier on games from 1998-2015 instead of only on games from 1998 to the year for which upsets were being identified was made due to the limited quantity of available historical data.

Finding a suitable set of statistics to use could have also been done by enumeration and validation, but the classifier was used to avoid enumerating $2^{115} - 1$ different potential combinations of statistics. None of the decision trees generated by the extra-trees classifier are being used in any way to select potential upsets.

3.3.3 Finding Similar Games

The next step of the technique is to use the set of 15 match-up statistics $\mathcal{S}$ to select potential upsets. The Balance Optimization Subset Selection (BOSS) framework (Nikolaev et al., 2013) is used to do the selection.

For selecting potential upsets, the control pool consisted of the 13, 14, and 15-seed round-of-64 games and the treatment group consisted of historical upsets. The control group selected by BOSS would be the set of match-ups most similar to the historical upsets according to the defined balance measure. For each combination of four match-up statistics $S \subset \mathcal{S}$, BOSS is used to select three match-ups as potential upsets. These will then be narrowed to two final selections in the next step of the technique.

BOSS requires a balance measure by which games in the current year can be compared to upsets in the past to measure the similarity between the control group and the treatment group. The balance measure used was a combination of (1) the difference between the empirical distribution of the treatment and control group for each statistic $i \in S$ and (2) the relative difference between the sum of each statistic for each game in the control and treatment group. The difference between the empirical distributions was measured using the Kolmogorov-Smirnov (KS) test statistic, which defines the distance K between two sets f_1 and f_2 with empirical distributions F_1 and F_2 respectively over a set of values V as

$$K(f_1, f_2, V) \equiv \max_{v \in V} |F_1(v) - F_2(v)|. \tag{18}$$

The Kolmogorov-Smirnov statistic measures the maximum vertical distance between two empirical distributions.

Since the KS statistic only measures the difference in the height of the empirical distributions, it is possible for the first and last values in the empirical distribution to be significantly further apart on one distribution than the other while retaining the same KS statistic value. For example, the KS statistic between {1, 2, 3, 4} and {1, 2, 3, 5} would be the same as the KS statistic between {1, 2, 3, 4} and {1, 2, 3, 1000}. In order to prevent that last value from being very far away from the rest of the distribution, we also use the relative difference between the distributions to include the horizontal difference between the two sets. The relative difference constraints force the horizontal spread to be similar in both distributions.

The relative difference R between the sum of the sets f_1 and f_2 is defined as:

$$R(f_1, f_2) \equiv \frac{|\frac{1}{|f_1|}\sum_{g \in f_1} g - \frac{1}{|f_2|}\sum_{g \in f_2} g|}{\frac{1}{|f_2|}\sum_{g \in f_2} g}. \tag{19}$$

The *covariates* in this problem are the different match-up statistics being used. Let the following terms be defined as:

- T: Treatment Group
- C: Control Pool
- G: Control Group
- $\mathbf{X}$: Set of covariates
- S_i: Values of set S for covariate i
- V_i: Set of unique values in $T \cup C$ for covariate $i \in \mathbf{X}$

We then set our balance measure $M(G)$ for control group G to be

$$M(G) = \sum_{x \in \mathbf{X}} \max\{K(T_i, G_i), R(T_i, G_i)\}. \tag{20}$$

We then find group G with size $|G| = 3$ such that $M(G)$ is minimized. The three teams in G will be the three selections for the set of covariates S. Due to the small size of both T and C, we solved BOSS by enumerating all possible sets of G and choosing the one with the smallest $M(G)$. BOSS can also be solved via a Mixed Integer Program (MIP), the formulation for which is presented in the appendix.

BOSS was run once for each combinations of four statistics out of the 15 chosen by the classifier (1365 combinations total) on years from 2001-2015 using each combination of statistics as covariates. The earliest year used was 2001 because detailed data for years prior to the 1997-1998 season were unavailable from TeamRankings.com and forming a treatment group requires historical upsets, so some years would have to be used to build a small treatment group. The upsets in years from 1998-2000 were used to form the treatment group for 2001.

3.3.4 Combining Models

Solving BOSS with each $S \in \mathcal{S}$ produced 1365 sets of three match-ups each (one set of three match-ups from each combination of four match-up statistics).In order to finally select two match-ups as potential upsets, the results of those 1365 BOSS solutions must be combined to yield two match-ups. In order to do this, a reasonable action is to take the two match-ups that appear most frequently across the set of BOSS solutions. However, not all combinations of match-up statistics are equally informative or valuable. Therefore, only those combinations of match-up statistics that proved to provide accurate solutions historically were included. The subset of combinations to use was chosen by evaluating the performance of each combination of match-up statistics using historical data and selecting the ones with the best past performance. The performance of a given combination of match-up statistics was measured as the number of upsets that BOSS selected correctly

over the entire range of years used when the given combination was used as covariates. To formalize this, let $n_{S,y}$ be the number of upsets in year y that were included in the BOSS solution when optimizing over the match-up statistics S (so $n_{S,y} \in \{0, 1, 2, 3\}$). Then let $N_{S,y} = \sum_{y'<y} n_{S,y'}$ be the historical performance of S in predicting upsets up through but not including year y. For any given year y, let $N_y^* = \max_{S \subseteq \mathcal{S}} N_{S,y}$ be the largest number of selected upsets across all match-up statistics. Then let $P_y = \left\{ S \in \binom{\mathcal{S}}{4} : N_{S,y} \geq N_y^* - \tau \right\}$ be the set of high-performing match-up statistics in year y, where τ is a tolerance parameter. The statistics in P_y are then used to select upsets for year y. The tolerance value τ was determined by testing values between one and 20 and choosing the value that yielded the most correct selections, where a correct selection is the selection of a game that was an actual upset. The value of τ was determined separately for each year.

The steps to select two teams for year Y from the results of BOSS are as follows:

1. Find the number of correct selections made by BOSS for each combination of four match-up statistics $S \in \binom{\mathcal{S}}{4}$ from year 2001 to year $Y-2$. Let the number of correct selections for S be $N_{S,Y-2}$.

2. Find the single combination of four match-up statistics for which BOSS made the most correct selections when run from year 2001 to year $Y-2$. Let this be N_{Y-2}^*.

3. Set τ to one. Find all combinations of match-up statistics for which BOSS selected at least as many upsets correctly as the number correctly selected by the best combination found in the previous step minus the tolerance value ($N_{Y-2}^* - \tau$) for years 2001 to $Y-2$. Let P_{Y-1} be the set of the selected combinations.

4. Use the two teams most frequently selected by BOSS for year $Y-1$ when run on each combination of match-up statistics in P_{Y-1} as the two selections for year $Y-1$.

5. Repeat steps (1) - (3) for each τ between one and 20. Select the value of τ that resulted in the most correct selections for year $Y-1$.

6. Use the value of τ found in step (4) to make selections for year Y using steps (1) - (3) but iterating to year $Y-1$ instead of $Y-2$ in step 2.

Algorithm 1 provides the pseudo-code for selecting the subset of combinations of match-up statistics to use and combining the teams chosen using each combination in that subset into the two selected games for a target year Y where Y is a year after 2002. Lines 3-12 iterate through each year prior to Y and determine the number of correct selections that are made for each potential tolerance value. Lines 5-8 find the number of correct selections made using each combination of match-up statistics by comparing the games selected by BOSS using that combination of match-up statistics to the actual historical upsets that occurred.

Algorithm 1 Generating selections for year Y using BOSS

1: $\mathbb{S} \leftarrow \{\text{all } S \subset \mathcal{S} : |S| = 4\}$
2: **for** $\tau \in \{1, 2, ..., 20\}$ **do**
3: $\quad g_\tau \leftarrow \{\}$
4: $\quad$ **for** year $y \in \{2002, 2003, ..., Y-1\}$ **do**
5: $\quad\quad$ **for** each $s \in \mathbb{S}$ **do**
6: $\quad\quad\quad N_{s,y} \leftarrow$number correct selections by s using BOSS for years $\{2001, .., y-1\}$
7: $\quad\quad$ **end for**
8: $\quad\quad N_y^* \leftarrow \max_{s \in \mathbb{S}} \{N_{s,y}\}$
9: $\quad\quad P_y \leftarrow \{s \in \mathbb{S} : N_{s,y} > N_y^* - \tau\}$
10: $\quad\quad g_\tau = g_\tau \cup \{$the two games most frequently selected by all $s \in P_y$ using BOSS for year $y\}$
11: $\quad$ **end for**
12: **end for**
13: $\tau_{opt} \leftarrow \tau$ such that g_τ contains the most correct upsets for years $\{2002,...,Y-1\}$
14: **for** each $s \in \mathbb{S}$ **do**
15: $\quad N_{s,Y} \leftarrow$number correct selections by s using BOSS for years $\{2001, .., Y-1\}$
16: **end for**
17: $N_Y^* \leftarrow \max_{s \in \mathbb{S}} \{N_{s,Y}\}$
18: $P_Y \leftarrow \{s \in \mathbb{S} : N_{s,Y} > N_Y^* - \tau_o pt\}$
19: return the two games most frequently selected by all $s \in P_Y$ using BOSS for year Y

The combinations that performed within the tolerance of the best single combination are used to generate the final two selections. The final selections are the two match-ups that are most selected by the chosen combinations. The tolerance value that results in the most accurate selections for years between years 2002 and $Y-1$ is then used to select two games as potential upsets for year Y in lines 14-20. If multiple tolerance values generated the same number of correct selections, the maximum of those tolerance values was used. In the event that there was a tie for the most frequently occurring team or the second most frequently occurring team, more than two teams would have been selected. However, such a tie never occurred, so this contingency was never used.

Combining the results of multiple models or instances of a model, known as *ensembling*, has been shown to frequently reduce errors due to a specific failing in individual models (Opitz and Maclin, 1999). Due to the small size of the dataset, one specific set of covariates may be high performing, but a high performance by a single covariate combination may be due to coincidence since the outcome of each game is essentially a random variable. Ensembling multiple models or instances of a model should make the resultant ensemble more resistant to overfitting the dataset, but determining the amount of overfitting is difficult due to the limited number of years for which there is historical team data.

3.4 Results

The technique presented in Section 3.3 was used to select two potential upsets for years between 2003 and 2015. The games selected as upsets are listed in Table 4. The number of upsets that occurred each year is shown in Table 6. Table 7 lists the selection frequency and accuracy for each seed separately.

In total, the presented technique selected 10 upsets correctly out of 26 picks (38.4%) over 13 years.

Analysis of the results lead to several observations about the tendencies of the technique. We selected two upsets correctly one time, one upset correctly eight times, and zero upsets correctly four times. However, out of the four years where we got zero correct, two years had zero upsets actually occur. Therefore, we selected at least one upset correctly in nine out of the 11 years that had at least one upset occur. Moreoever, given that we choose exactly two upsets per year, we can observe from the historical record that the maximum number of upsets that we could have chosen correctly is 18. Therefore, we selected 10 out of the 18 possible upsets that we could have selected correctly. The games chosen also tend to favor stronger seeds, as we pick a 13-seed fourteen times, a 14-seed eight times, and a 15-seed four times. Selecting higher-seed teams with higher frequency is reasonable since the 13-seeds are more likely to win than the 15-seeds.

Another interesting observation is that the tolerance value determined when using Algorithm 1 for each year remained constant for all years from 2009 onward. A constant tolerance could be evidence of some level of stability, since the fact that it stayed constant for seven consecutive years suggests that is likely to be the correct value to use in future years. However, due to the limited number of years of data available, the value for further years should be determined using the algorithm until this theory can be further tested.

To compare the performance of our technique to predictions made by randomly choosing games, we determined the expected number of correct selections when two teams were randomly selected as predicted upsets. We can either randomly select two teams each year with equal probability or, since we know the historical frequency of each seed winning a round-of-64 game, we can randomly select two possible upsets each year using the historical frequency of an upset occurring for each seed as weights. The weights used for weighted random selection each year were calculated using upsets that occurred prior to that year. For example, when randomly selecting teams as predictions for 2010, the frequency of upsets from 1985 to 2009 was used for weighting. Each game was then determined to be an upset by modeling it as a Bernoulli variable with the probability being the fraction of historical games of that seed match-up that resulted in upsets. The following proposition establishes the number of upsets that would be correctly predicted through (1) random selection where each team has the same probability of being selected and (2) random selection where the probability of each team being selected is weighted based on the historical frequency of that team's seed resulting in an upset.

In order to compute the expected value and variance, let the following terms be defined:

- U_y: Random variable representing the number of upsets selected correctly in year y
- x_{sy}: Number of games that seed s has won before year y
- $G_{s,y}$: Number of round-of-64 games played by s-seed teams before year y
- n_y: Total number of upsets that occurred prior to year y.

The variables are dependent on the year y because the weighted probabilities used each year are computed by using the frequency of upsets occurring before that year. Therefore, these probabilities change each year as new upsets occur each year of the tournament.

Proposition 1. *When choosing two 13, 14, or 15-seeded match-ups as upsets randomly for each year between y_1 and y_2 where the probability of choosing each team is the historical frequency with which upsets occur for that seed, the expected number of upsets selected correctly is*

$$\begin{aligned} E[U_{[y_1,y_2]}] = \sum_{y\in[y_1,y_2]} \Bigg(& 4\left(\frac{x_{sy}}{4n_y} + \frac{3x_{sy}}{4n_y}\frac{x_{sy}}{4n_y - x_{sy}}\right) \\ & + 4 \sum_{i\in\{13,14,15\}:i\neq s} \frac{x_{iy}}{4n_y}\ \frac{x_{sy}}{4n_y - x_{iy}}\Bigg)\left(\frac{x_{sy}}{G_y}\right), \end{aligned} \tag{21}$$

and the variance is

$$\begin{aligned} Var[U_{[y_1,y_2]}] = \sum_{y\in[y_1,y_2]} \Bigg(& 4\left(\frac{x_{sy}}{4n_y} + \frac{3x_{sy}}{4n_y}\frac{x_{sy}}{4n_y - x_{sy}}\right) \\ & + 4 \sum_{i\in\{13,14,15\}:i\neq s} \frac{x_{iy}}{4n_y}\ \frac{x_{sy}}{4n_y - x_{iy}}\Bigg)\left(\frac{x_{sy}}{G_y}\right) \\ & + 8\Bigg(\sum_{s_i=s_j} \frac{x_{s_iy}}{4n_y}\frac{3x_{s_iy}}{4n_y - x_{s_iy}}\left(\frac{x_i}{G_y}\right)^2 \\ & + \sum_{s_i\neq s_j} \frac{x_{s_iy}}{4n_y}\frac{4x_{s_jy}}{4n_y - x_{s_iy}}\frac{x_i x_j}{G_y^2}\Bigg). \end{aligned} \tag{22}$$

Proof. The following equations will compute the expected value and variance when using weighted random selection. A modification to use uniform random selection is provided at the end of the proof.

The probability of randomly selecting a specific team with seed s is

$$\begin{aligned} P(\text{selecting team with seed } s) = & P(\text{choose team first}) \\ & + P(\text{not choose first and choose second}) \end{aligned} \tag{23}$$

Since there are four teams with each seed, this probability is multiplied by four. Also, the probability depends on year y. Therefore,

$$\begin{aligned} P(\text{selecting team with seed } s \text{ in year } y) = 4 \left(\frac{x_{sy}}{4n_y} + \frac{3x_{sy}}{4n_y} \frac{x_{sy}}{4n_y - x_{sy}} \right) \\ + 4 \sum_{i \in \{13,14,15\}: i \neq s} \frac{x_{iy}}{4n_y} \frac{x_{sy}}{4n_y - x_{iy}}. \end{aligned} \tag{24}$$

The probability of the selected team being an actual upset is

$$P(\text{seed } s \text{ correct in year } y) = \frac{x_{sy}}{G_y}. \tag{25}$$

Therefore for each year, the expected number of correctly predicted upsets is

$$\begin{aligned} E[U_y] = \sum_{s \in \{13,14,15\}} P(\text{selecting team with seed } s \text{ in year } y) \\ * P(\text{seed } s \text{ correct in year } y) \end{aligned} \tag{26}$$

Since the results of each year are independent, as upsets occurring one year do not depend on upsets occurring in the previous years, we can add the expected number of upsets each year to arrive at the expected number of upsets over a range of years. When x_s, n, and G are determined by historical data prior to each year, we find the expected number of upsets from y_1 to y_2 (including y_2) to be

$$E[U_{[y_1,y_2]}] = \sum_{y \in [y_1,y_2]} E[U_y]. \tag{27}$$

To compute the variance, we can rewrite the expected number of upsets as

$$E[U_y] = P(U_y = 1) + 2P(U_y = 2). \tag{28}$$

To compute the variance, $E[U_y]^2$ is required.

$$E[U_y]^2 = 1^2 * P(U_y = 1) + 2^2 * P(U_y = 2) = E[U_y] + 2P(U_y = 2) \tag{29}$$

To find $P(U_y = 2)$, we let (s_i, s_j) be all possible seed pairs where i and j are each drawn from {13, 14, 15}

with replacement . Then

$$P(U_y = 2) = 4\left(\sum_{s_i=s_j} \frac{x_{s_i y}}{4n_y}\frac{3x_{s_i y}}{4n_y - x_{s_i y}}\left(\frac{x_i}{G_y}\right)^2 + \sum_{s_i \neq s_j} \frac{x_{s_i y}}{4n_y}\frac{4x_{s_j y}}{4n_y - x_{s_i y}}\frac{x_i x_j}{G_y^2}\right). \tag{30}$$

This allows us to express the variance as

$$\begin{aligned} Var[U_y] = E[U_y] + 8\Bigg(&\sum_{s_i=s_j} \frac{x_{s_i y}}{4n_y}\frac{3x_{s_i y}}{4n_y - x_{s_i y}}\left(\frac{x_i}{G_y}\right)^2 \\ &+ \sum_{s_i \neq s_j} \frac{x_{s_i y}}{4n_y}\frac{4x_{s_j y}}{4n_y - x_{s_i y}}\frac{x_i x_j}{G_y^2}\Bigg). \end{aligned} \tag{31}$$

Since the result of each year is independent, we can say that

$$Var[U_{[y_1,y_2]}] = \sum_{y \in [y_1,y_2]} Var[U_y]. \tag{32}$$

The above equations express the expected number and variance of correct selections when using weighted random selection. In order to compute the expected number and variance of correct selections when each seed is equally likely to be chosen, modify (24) to be

$$P(\text{select team with seed } s \text{ in year } y) = 4\left(\frac{1}{12} + \frac{11}{12}\frac{1}{11}\right) = \frac{8}{12}, \tag{33}$$

and (30) to be

$$P(U_y = 2) = \sum_{s_i=s_j} \frac{4}{12}\frac{3}{11}\left(\frac{x_i}{G_y}\right)^2 + \sum_{s_i \neq s_j} \frac{4}{12}\frac{4}{11}\frac{x_i x_j}{G_y^2}. \tag{34}$$

with the other equations suitably modified since we want each seed to be chosen with probability 1/3 instead of having them depend on the historical frequency of upsets by each seed. □

By using available historical data and uniform random selection and the result from the proposition, the expected number of upsets to be chosen correctly when two upsets are selected per year between 2003 and 2015 is 3.26 with a variance of 2.93. These values change to 4.42 and 3.36, respectively, when using weighted random selection. The year-by-year expected value and variance for weighted random selection can be found in table 5.

Our technique selected 10 upsets over the 13 year period between 2003 and 2015. Therefore, our technique produced a number of correct selections that is 2.92 standard deviations above the expected number of correct predictions if weighted random selection were used. If uniform random selection is used, our technique

produces a number of correct selections that is 3.94 standard deviations above the expected value of 3.26 correct selections. This is a key comparison to establish the performance of our technique, since what we observe is that our technique performs significantly better than if we used a form of random selection. This means that using our technique to select potential upsets is a more reliable way of identifying potential upsets than choosing match-ups randomly.

We also applied our technique to the 11 and 12 seeded games instead of the 13, 14, and 15 seeded games. For the 11 and 12 seeded games, we modified Algorithm 1 to select three teams instead of two. Table 8 provides the games selected. When run on the 11 and 12 seeds, there were cases where we had ties; the last step of our technique found multiple games that were selected with the same frequency. In order to resolve this, we count the number of correct selections by weighting each correct selection by the frequency with which it would be chosen if the ties were resolved by random selection. For example, if two teams were tied for third-most-selected and one of them was a correct upset selection, those two teams would be combined into a score of 0.5. In the event of a three-way tie for second with one correct selection, the resulting score would be 1/3. However, the accuracy of the model for the 11 and 12 seeded games was comparable to that expected by weighted random selection. If weighted random selection were used, the expected number of correct selections would be 12.71 with a variance of 8.56, while our technique selected 10.67 upsets correctly. One hypothesis as to why this might be the case is that there is enough information that can be drawn from the covariates of upsets in the past that makes it possible to predict upsets in the future better than randomly selecting teams for 13-15 seeded games, while the 11 and 12 seeded games do not contain enough distinguishing information in the statistics available to us. Since the gap between the seeds in the 11 and 12 seeded games is not at large, the inherent randomness of the games might be overwhelming the information that the covariates provide about what causes an upset to occur for those seeds.

In order to determine the computational complexity of our technique, each step of the technique was run 10 times on a computer with an Intel Xeon E3-1246 quad-core processor at 3.5ghz with 16GB of memory. The runtimes for each step are listed in table 9.

In order to further evaluate the efficacy of our technique, we compare it to other methods found in the literature. We compared our results to from the technique described in Lopez (2015), which predicts the probability of each team winning against each other team. Therefore, to make a fair comparison, we used the Lopez (2015) model to generate the probabilities of each low seed winning their first-round game and selected the two games where the lower seeded team had the highest probability of winning. The model was trained separately for each year using all games that occurred prior to that year's tournament. One note is that their model uses the home team stats and away team stats as inputs to their logistic regression. In the event of a neutral game, we randomly assigned one team as home and one team as away in addition

to marking the game as neutral using the neutral indicator in their model. Due to this randomization, we ran their model 100 times and for each run counted how many upsets would have been selected correctly. Their model gives an average of 7.46 upsets correctly out of 26 selections, with a maximum of nine and a minimum of six. Figure 13 shows a histogram of the number of upsets correctly by the Lopez technique. Since our technique selected 10 upsets correctly out of 26 selections, our technique was strictly dominant over the given time period.

Bryan, Steinke, and Wilkins (2006) did an analysis on predicting round-of-64 upsets using a regression model where they define an *upset* as a game where the lower-seeded team wins and a *nonupset* as a game where the higher-seeded team wins. Their results were that 41.8% of the games they selected as upsets were actually upsets and 80.99% of the games they selected as nonupsets were actually nonupsets between 1994 and 2005 and 36.36% of the games they selected as upsets were actually upsets and 80.26% of the games they selected as nonupsets were actually nonupsets between 2000 and 2005. However, they declared their model as successfully predicting an upset if "it predicts a probability of upset greater than the historic proportion of games at the given seed difference that resulted in an upset". They also considered upsets as a 10, 11, 12, or 13-seed winning a game, whereas we consider an upset as a 13, 14, or 15-seed winning. Since we choose games specifically as upsets rather than those where the weaker team is more likely to win than the historical average and have a different definition for what constitutes an upset, the results are not directly comparable.

3.5 Conclusions

This paper presents a technique to select round-of-64 NCAA tournament upsets using game statistics. The technique identifies important statistics and uses those statistics to find match-ups similar to historical upsets. The performance of the technique was tested by using the technique to select potential upsets for the years 2003-2015. The technique was shown to significantly outperform random selection, both when the random selection was done with a uniform random distribution and when the distribution was weighted by the historical frequency of each seed winning a round-of-64 game.

There are several limitations to the technique used in this paper. First, the identification of important match-up statistics was done using all the years of data rather than identifying the important statistics using only the data that were available in each year. This was done due to the limited amount of data available; however, in the future there will be enough data for the choice of statistics to not vary from year to year. Furthermore, the technique as presented is limited to choosing a fixed number of potential upsets each year. This is a limitation because historical data shows that the number of upsets that occurs in a year can vary

from zero to three, and this technique does not account for that. BOSS could be adapted to identify the most likely upset or the three most likely upsets by changing the desired size of the control group; additional modifications would be needed for BOSS to decide whether no upsets should occur. Finally, due to the limited number of years for which we have data, it is difficult to optimize the parameters of the technique. Since the number of upsets being selected is so small, even a small variation in the number of upsets selected correctly presents itself as a large change in the accuracy percentage. This means that some of the parameters chosen might not be optimal, but rather happened to perform slightly better on the small amount of data we had. As more years of data become available, the technique will be able to be tested more thoroughly.

Some potential areas for future work are experimentation with different match-up statistics, different methods for matchup-statistic combination selection (namely modifications to algorithm 1), and the adaptation of the technique to be able to select a varying number of games as potential upsets. The primary improvement for this technique, however, will come with the availability of more data that will allow for further experimentation and testing. More data will allow the determination of accuracy to be more robust and less sensitive to each individual upset. Given the small size of the control group, another direction for future work is to develop efficient algorithms for solving BOSS directly without the use of MIP models. This paper enumerated through all combinations of possible control groups as a substitute to solving the MIP due to the small size of the data, but on larger problems alternative methods could be useful.

3.6 Figures and Tables

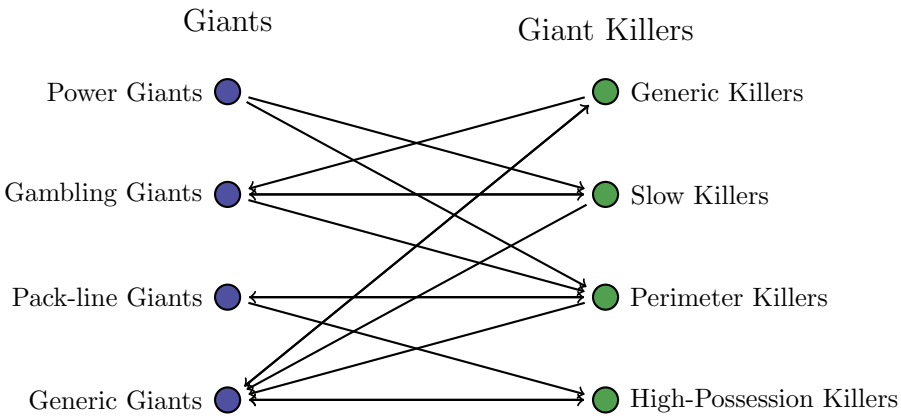

Figure 12: Which Giants might lose to which Giant Killers and which Giant Killers might win against which Giants

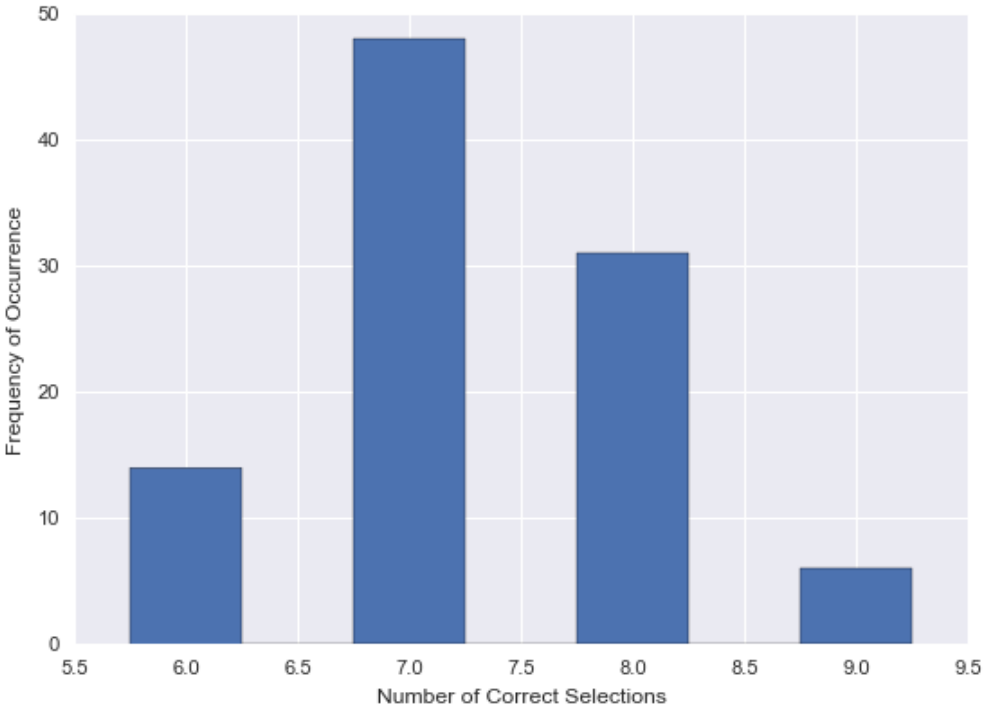

Figure 13: Frequency of number of correct selections by Lopez (2015) model

Year	Game Seed	Winning Team	Losing Team	Upset Selected Correctly
2015	14	Georgia St	Baylor	True
2015	14	UAB	Iowa State	True
2014	13	San Diego St	N Mex State	False
2014	13	Michigan St	Delaware	False
2013	14	Marquette	Davidson	False
2013	13	La Salle	Kansas St	True
2012	14	Georgetown	Belmont	False
2012	15	Lehigh	Duke	True
2011	15	N Carolina	LIU-Brooklyn	False
2011	15	Notre Dame	Akron	False
2010	14	Ohio	Georgetown	True
2010	15	W Virginia	Morgan St	False
2009	13	Xavier	Portland St	False
2009	13	Cleveland St	Wake Forest	True
2008	13	Siena	Vanderbilt	True
2008	13	Pittsburgh	Oral Roberts	False
2007	13	S Illinois	Holy Cross	False
2007	13	Virginia	Albany	False
2006	14	Gonzaga	Xavier	False
2006	13	Bradley	Kansas	True
2005	14	Oklahoma	Niagara	False
2005	13	Vermont	Syracuse	True
2004	13	Maryland	TX El Paso	False
2004	13	Kansas	IL-Chicago	False
2003	13	Tulsa	Dayton	True
2003	14	Xavier	Troy	False

Table 4: Games Selected

Year	Expected Number Upsets Selected Correctly	Variance
2015	0.331	0.276
2014	0.337	0.280
2013	0.335	0.278
2012	0.343	0.284
2011	0.344	0.285
2010	0.339	0.281
2009	0.341	0.282
2008	0.330	0.276
2007	0.345	0.286
2006	0.340	0.281
2005	0.333	0.277
2004	0.35	0.289
2003	0.354	0.291

Table 5: Number of Randomly Selected Upsets Correctly Chosen

Year	Number of Actual Upsets	Number Selected Correctly
2015	2	2
2014	1	0
2013	3	1
2012	3	1
2011	1	0
2010	2	1
2009	1	1
2008	2	1
2007	0	0
2006	2	1
2005	2	1
2004	0	0
2003	1	1

Table 6: Number of Upsets per Year

Seed	Number of Actual Upsets	Number Selected	Number Selected Correctly
13	10	14	6
14	7	8	3
15	3	4	1
Total	20	26	10

Table 7: Selection Accuracy by Seed

Year	Game Seed	Winning Team	Losing Team	Upset Selected Correctly
2015	11	UCLA	S Methodist	True
2015	11	Butler	Texas	False
2015	12	Utah	Ste F Austin	False
2014	12	Saint Louis	NC State	False
2014	11	N Carolina	Providence	False
2014	12	N Dakota St	Oklahoma	True
2014	11	Tennessee	U Mass	True
2013	11	Arizona	Belmont	False
2013	11	Memphis	St Marys	False
2013	12	Oregon	Oklahoma St	True
2012	12	New Mexico	Lg Beach St	False
2012	11	Cincinnati	Texas	False
2012	11	NC State	San Diego St	True
2011	12	Richmond	Vanderbilt	True
2011	12	Arizona	Memphis	False
2011	11	Cincinnati	Missouri	False
2010	11	Tennessee	San Diego St	False
2010	11	Old Dominion	Notre Dame	True
2010	12	Cornell	Temple	True
2009	12	W Kentucky	Illinois	True
2009	11	Marquette	Utah State	False
2009	11	UCLA	VCU	False
2008	12	Notre Dame	Geo Mason	False
2008	11	Oklahoma	St Josephs	False
2008	12	W Kentucky	Drake	True
2007	12	USC	Arkansas	False
2007	12	Butler	Old Dominion	False
2007	11	Louisville	Stanford	False
2007	11	Vanderbilt	Geo Wshgtn	False
2007	11	Winthrop	Notre Dame	True
2006	11	WI-Milwkee	Oklahoma	True
2006	11	Indiana	San Diego St	False
2006	12	Washington	Utah State	False
2005	12	GA Tech	Geo Wshgtn	False
2005	12	Villanova	New Mexico	False
2005	11	Texas Tech	UCLA	False
2004	12	Illinois	Murray St	False
2004	12	Syracuse	BYU	False
2004	11	Vanderbilt	W Michigan	False
2003	11	Maryland	NC-Wilmgton	False
2003	11	Oklahoma St	U Penn	False
2003	11	Missouri	S Illinois	False
2003	12	Butler	Miss State	True

Table 8: Games Selected

Step	Min (s)	Mean (s)	Max (s)
Compute match-up statistics	1	1	1
Identify useful match-up statistics	138	154	170
Find similar match-ups	2331	2425	2607
Combine models	25	27	30

Table 9: Runtimes for Each Step of Technique

4 Modeling the NCAA Basketball Tournament Selection Process

4.1 Introduction

The men's college basketball championship tournament held by the National Collegiate Athletic Association (NCAA), henceforth referred to as *the tournament*, attracts mass media and popular attention across the country. Surprisingly, for such a popular event, the method by which teams are selected for participation in the tournament is something of a mystery. Each year 32 teams are guaranteed entry by winning their conference tournaments. However, the other 36 teams are selected by a selection committee whose job is to determine which teams are best qualified to play in the tournament. While the procedure followed by the selection committee to select the 36 teams is known, the details on exactly how the teams are ranked have not been made fully public. There has been much speculation on which factors are important to the committee, how quantitative or qualitative their assessment is, and how much of it is just the "eye test".

In this paper we propose a decision-making method that attempts to mimic the process performed by the selection committee. Here we focus solely on the selection process, where we identify the teams that will be given entrance into the tournament, rather than also attempting to predict the seed each team will be assigned. We then present the results of our method when tested over the years 2011-2016, and show that we are able to correctly select all but one team in each of those years.

4.2 Background

There has been much public speculation on how the committee selects teams for the tournament. The process by which a committee selects teams is public (NCAA, 2016); each committee member identifies 36 teams which they believe should be given entrance, as well as marks any team which they believe should be considered for entry. Then, any team which at least all-but-two committee members select for inclusion will be added to the tournament. The remaining teams are taken from the lists for inclusion and consideration and filtered down via another series of ballots. Finally, those teams are ranked by each committee member, and the highest ranked teams are added to the tournament four at a time.

The uncertainty comes when attempting to determine how a committee member decides which teams are stronger than other teams. The members have copious amounts of data available, including box scores, head-to-head results, results against common opponents, conference schedules, overall and non-conference strength of schedule, road record, injury reports, coach availability, and other tracked metrics (NCAA, 2016). However, it is not known which of these pieces of information is used and how much they are weighted. For example, it was long thought that the Rating Performance Index (RPI) was a large factor in the selection

process (Ezekowitz, 2013), but recent statements from committee members state that the RPI is used to determine which teams are considered, but is not a determining factor (Stephens, 2015). Other statements by committee members suggest the importance of acquiring top-50 wins (wins against a team with a top-50 RPI) and avoiding 200+ losses (losses against a team with an RPI of over 200) (Katz, 2016).

The RPI metric is used to rank teams based on each team's wins, losses, and strength of schedule (how strong their opponents were). The current formula for RPI is given as

$$RPI = (WP * 0.25) + (OWP * 0.5) + (OOWP * 0.25) \tag{35}$$

where WP is Winning Percentage, OWP is Opponents' Winning Percentage, and OOWP is Opponents' Opponents' Winning Percentage. The WP is calculated as number of wins / number of games, where a home win is counted as 0.6, a neutral win is 1, and an away win is 1.4. This weighting is due to the observed influence of game location on game outcome, where home teams tend to win significantly more than away teams. The OWP is the average of the WP for each of a team's opponents, while the OOWP is the average of each opponent's OWP.

While official statements may seek to downplay the role of RPI, the RPI values are embedded into numerous pieces of information the committee admits to using. For example, top-50 wins are counted as wins against teams with the top 50 RPI ranks, while strength of schedule measure the strength of each team using it's RPI rank. Therefore, the RPI of each team is inherently included in the committee selection process.

The RPI has also been subjected to criticisms. One of the most frequent criticisms is that it relies too heavily on strength of schedule (Pomeroy, 2011). As seen in the formula for RPI, the strength of the opponents appears in both the OWP and OOWP components, which means that 75% of the RPI value is dependent on opponent strength to some degree. This can cause problems for teams in weaker conferences, since they do not play against strong opponents in their conference. In fact, playing a weaker team can be detrimental to a team's RPI rank regardless of the outcome of the game. Another criticism is that RPI does not take into account margin of victory. This is ostensibly to discourage the manipulation of point outcomes in the context of gambling, but results in ignoring potentially useful information. However, the RPI rankings are officially acknowledged to be used in the selection process, so we chose to include them.

In light of the weaknesses found in the RPI, several other rating systems have risen to popularity. The Basketball Power Index (BPI) was introduced in 2013 by ESPN and referred to as "a little more refined than any other existing power ranking"(Oliver, 2013). The formula for the BPI is not included in the literature, but it is said to include information such as whether a team is missing an important player during a game,

how close the game was, the pace of the game, and strength of schedule (Oliver, 2013).

Another popular rating system is provided by Ken Pomeroy, who scores each team using a *pythagorean winning percentage* (Pomeroy, 2012). His ratings combine the *adjusted offensive efficiency (AdjO)* and *adjusted defensive efficiency (AdjD)* which are an estimate of the points scored (for offense) or allowed (for defense) per 100 possessions against the average defense (or offense). The formula to combine these into the pythagorean rating is

$$\text{pyth} = \frac{AdjO^{10.25}}{AdjO^{10.25} + AdjD^{10.25}} \tag{36}$$

Jeff Sagarin also publishes the popular Sagarin rankings, but his formula is also proprietary.

4.3 Methodology

The goal of this method is to predict which teams will be selected for entrance into the tournament. In order to do this, we must first choose which pieces of information pertaining to each team to use. This information will then be built into a model to determine the relative strength of the teams considered for entry in the tournament.

4.3.1 Choosing Relevant Information

We decided to use some expert rankings in conjunction with some performance metrics for the teams over the regular season.

The first included ranking is RPI, which is known to be a factor that the selection committee at least considers. We also chose to include other expert rankings to potentially make up for some of the weaknesses in the RPI system. The main expert rankings in popular use are the BPI, Pomeroy rankings, and the Sagarin rankings. In order to determine which ranking system to use, we compared the RPI, BPI, Pomeroy, and Sagarin rankings for each team between the years 2012 and 2016. The plots showing the pairwise comparison between these ranking systems can be found in figure 14, where each point on each plot represents the different rankings of a single team. We then calculated the pair-wise Pearson correlation coefficients between the RPI, BPI, Pomeroy, and Sagarin rankings (given in table 10) and observed that the BPI, Sagarin, and Pomeroy rankings were all highly correlated. Since the correlation coefficient was so large, using all of the ranking systems would provide only marginally more information than using just one; therefore, for the sake of simplicity, we opted to only use the Pomeroy rankings.

Choosing the relevant performance metrics to use was done by examining the statements made by the selection committee in the past. The metrics most frequently mentioned were the number of top 50 wins

(wins against the teams with the 50 highest RPI values) (Katz, 2016), the number of 200+ losses (losses against teams with an RPI rank of more than 200), and the strength of schedule of each team (NCAA, 2016).

Through experimental results, we also included the number of games played against the top 100 teams and the record of the team in their last 12 games before the tournament. The intuition behind including the last 12 games is that a team that plays well during their most recent 12 games are more likely to look strong to a selection committee than a team who plays poorly in those last games.

4.3.2 Selecting Tournament Teams

Selecting the teams for the tournament was a two-step process. The first step was to evaluate the teams in a pair-wise fashion, thereby determining which team was stronger than which other teams. To do this, each team was compared to each other team in the same year, and the stronger of the two teams (according to our evaluation method) was given one victory. Then, the teams with the most pair-wise victories were chosen for the tournament. The teams we used as candidates for tournament selection were the teams in the NCAA tournament with a seed of ten or higher who had been granted an at-large bid into the tournament and the one and two-seeded teams in the National Invitation Tournament (NIT), which is a tournament played by teams who were not granted entrance to the NCAA tournament. The best teams that played in the NIT should be a reasonable approximation for teams that were close to but did not get invited to the NCAA tournament. We used these teams because we decided to focus solely on the teams that may or may not make it into the tournament, rather than including the high-performing teams that are fairly certain to make the tournament. Our evaluation method for determining which team was the stronger of a given pair is modeled as a decision tree. The decision tree was created experimentally by trying to find a tree such that accurately modeled the selection process while maintaining simplicity. The goal was to create a tree without relying on complex conditions, since increasingly complex conditions could result in the tree being less generalizable and applicable to future years. Also, since the tree is meant to mimic the process done by the selection committee, it seemed unlikely that the conditions should be complex. The full tree is shown in figure 15, and the intuition behind the different conditions can be found in table 11.

Some of the data is preprocessed before being run through the tree. Instead of the raw strength of schedule value, we use the minimum of the strength of schedule and the non-conference strength of schedule. A strong non-conference strength of schedule indicates a team that made an effort to play against strong teams despite being in a weak conference. However, the original strength of schedule is also included so as not to penalize teams in strong conferences who do not feel the need to look elsewhere for strong competition. Additionally, the RPI, strength of schedule, and Pomeroy rankings are converted to ordinal values by year instead of using the raw values. This way, we avoid inter-year variation in the values, since each team is

ranked only against teams in the same year. Finally, the last 12 games of each team were combined into a single number, which was computed as the number of wins - the number of losses in those last 12 games. For example, A team with a 7-5 record in its last 12 games would be assigned a score of 7-5=2.

4.4 Results

Evaluating the accuracy of the tree was done by attempting to select the correct teams for the tournament for the years 2012-2016. The teams selected for each year and their rank is given in tables 12 - 16. The tables show the teams we used, the rank our method assigned to each team, and their seed in the tournament if they were granted entrance. In each of the five years, our method allowed one team into the tournament that did not actually make the tournament, but selected all the others correctly.

Some interesting observations can be made about the results. First, there was no single year where our method selected every team correctly. Whenever the method was adapted get every team right for a given year, it caused the other years to become significantly less accurate. Furthermore, some teams were entered into the tournament despite a seeming lack of foundation in the metrics used. Iona in 2012 is an example of this, with numbers that are far worse than other teams that were denied entrance to the tournament. This leads to the conclusion that the committee selection process is not an exact science - there seems to be some human factor involved that uses information outside the available numbers.

We can estimate which factors are most important to the committee by examining our decision tree. Factors used in a condition early on in the tree are more discriminatory, since they are able to operate with less prior information. In this tree, we can see that the RPI is highly distinguishing - it appears in two of the top three conditions. It is also implicitly present in the third one by virtue of the fact that top 50 and 200+ teams are determined by their RPI rank. Therefore, despite the statements fo the committee that RPI is not heavily used for selection, it seems to be either very important or a very good indicator. The RPI combined with the top 50 wins and 200+ losses seem to provide a strong indicator of whether a team will be in the tournament.

4.5 Conclusion

In this paper we presented a decision-making method that would mimic the results of the selection committee. We found that we could provide a close approximation, but that the actual selection process seems to also rely on factors that cannot be easily explained by numbers. In particular, we tested our method over the years 2012-2016, and found that it could selected all but one of the teams correctly each year. However, the remaining team was sometimes ranked far below others in our analysis.

One interesting avenue for future work would be to expand the method to incorporate seeding as well as selection. Here, we do not ascribe any value to the ranking of teams - they are either chosen for inclusion in the tournament or they are not. However, the real selection committee assigns each team a seed after the selection process is complete, so this would be a suitable addition to this method. Another possible area would be to incorporate more detailed information such as injury reports, further sources of expert opinion, etc. This would reduce the simplicity or the resultant model and therefore may be undesirable, but may also provide more accurate results. The largest gain, however, would come from more statements by the selection committee on exactly what information they use and value.

4.6 Figures and Tables

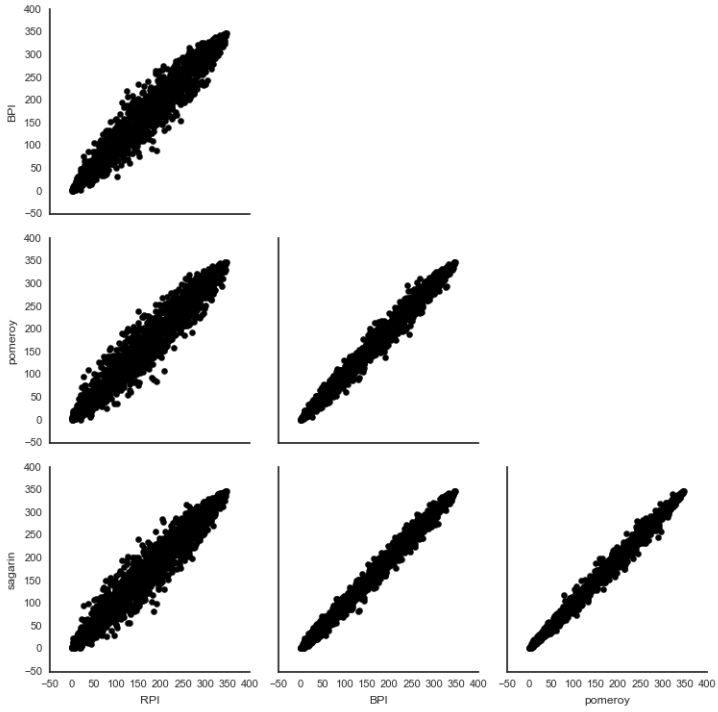

Figure 14: Plot comparing RPI, BPI, Pomeroy, and Sagarin

	RPI	BPI	Pomeroy	Sagarin
RPI				
BPI	0.967			
Pomeroy	0.965	0.991		
Sagarin	0.968	0.994	0.994	

Table 10: Correlation between ranking systems

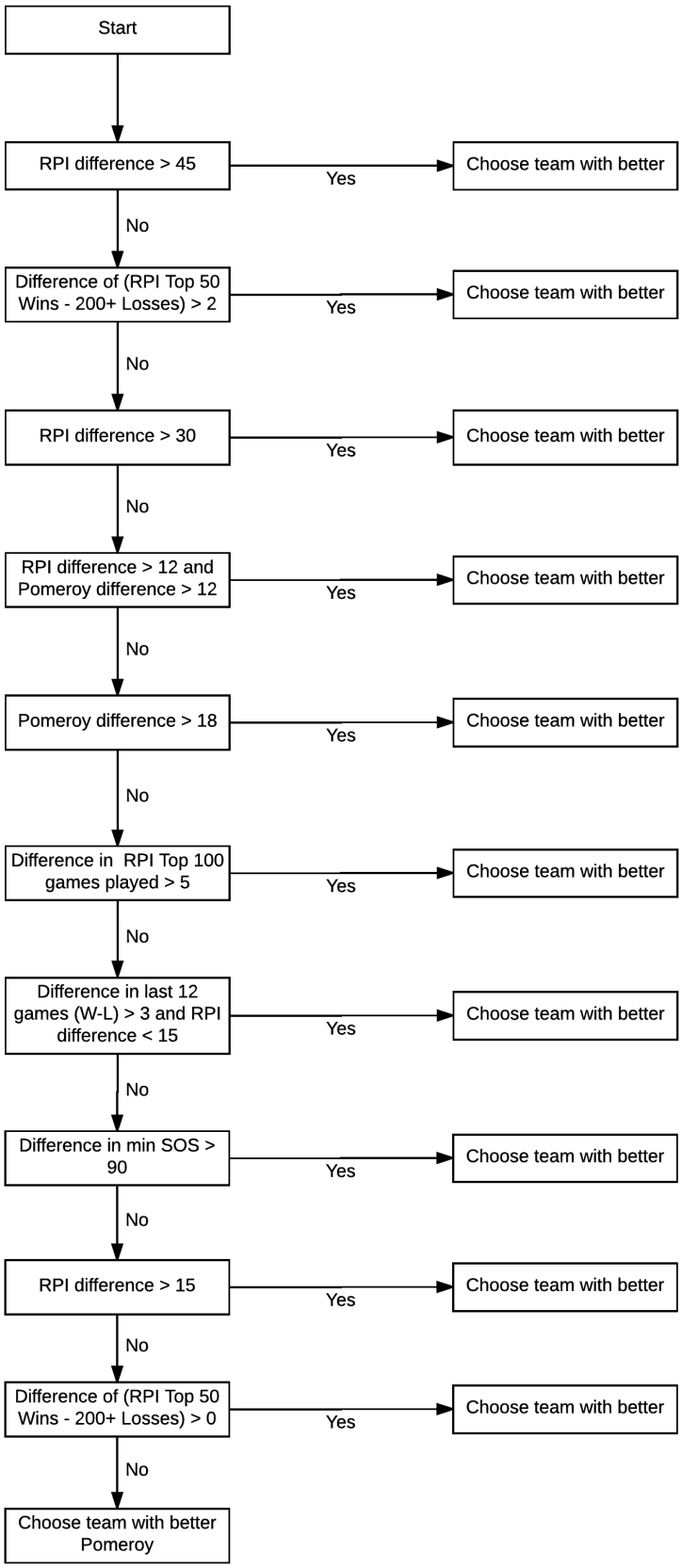

Figure 15: Tree to determine stronger of two teams

Condition	**Intuition**
RPI difference > 45	If one team has a much better RPI, choose that team
Difference of (RPI Top 50 Wins - 200+ Losses) > 2	One top 50 win is negated by one 200+ loss. If one team has at least 2 more, pick that team
RPI difference > 30	Same RPI condition with smaller threshold
RPI difference > 12 and Pomeroy difference > 12	If one team has both a moderately better RPI and Pomeroy ranking, pick that team
Pomeroy difference > 18	If one team has a moderately better Pomeroy ranking, pick that team
Difference in RPI Top 100 games played > 5	If one team has played at least 5 more games against the top 100 teams, pick that team
Difference in last 12 games > 3 and RPI difference < 15	If both teams have a similar RPI, pick the one that has done better in the past 12 games
Difference in min SOS > 90	If one team has had a much stronger strength of schedule, pick that team
RPI difference > 15	If one team has a moderately better RPI, pick that team
Difference of (RPI Top 50 Wins - 200+ Losses) > 0	Same as the second condition, but with a lower threshold
Choose team with better Pomeroy	If a decision has not been made yet, default to the team with better Pomeroy rank

Table 11: Correlation between ranking systems

Rank	School	Seed
1	Texas	11
2	Purdue	10
3	Xavier	10
4	South Florida	12
5	West Virginia	10
T6	NC State	11
T6	Virginia	10
T8	California	12
T8	Seton Hall	
11	BYU	14
12	Tennessee	
T13	Arizona	
T13	Drexel	
15	Saint Joesph's	
T16	Iona	14
T16	Oregon	
T16	Stanford	
19	Washington	
20	Dayton	

Table 12: 2012 Tournament Selections

Rank	School	Seed
1	Colorado	10
2	Iowa State	10
3	Minnesota	11
4	California	12
T5	Boise State	13
T5	Cincinatti	10
7	Saint Mary's	11
8	Baylor	
9	La Salle	13
10	Oklahoma	10
T11	Iowa	
T11	Middle Tennessee	11
13	Virginia	
14	Maryland	
15	Tennessee	
16	Kentucky	
T17	Massachusetts	
T17	Southern Miss	
19	BYU	
T20	Alabama	
T20	Arizona State	
22	Denver	

Table 13: 2013 Tournament Selections

Rank	School	Seed
1	Tennessee	11
2	Dayton	11
3	Nebraska	11
4	Stanford	10
5	Arizona State	10
T6	BYU	10
T6	Iowa	11
8	Xavier	12
9	Florida State	
10	SMU	
11	Minnesota	
T12	Arkansas	
T12	Illinois	
T12	Louisiana Tech	
15	NC State	12
T16	California	
T16	Southern Miss	
18	St. John's	
19	Missouri	
T20	Clemson	
T20	Georgia	

Table 14: 2014 Tournament Selections

Rank	School	Seed
1	Texas	11
2	Ohio State	10
3	UCLA	11
T4	BYU	11
T4	Indiana	10
T4	Ole Miss	11
T4	Temple	
8	Davidson	10
9	Boise State	11
10	Dayton	11
T11	Georgia	10
T11	Richmond	
13	Colorado State	
14	Illinois	
15	Stanford	
16	Rhode Island	
T17	Old Dominion	
T17	Texas A&M	
19	Tulsa	
20	Murray State	

Table 15: 2015 Tournament Selections

Rank	School	Seed
1	Syracuse	10
T2	Florida	
T2	Vanderbilt	11
T2	Wichita St	11
T5	Michigan	11
T5	Tulsa	11
T7	Pittsburgh	10
T7	VCU	10
9	Saint Mary's	
T10	St Bonaventure	
T10	Temple	10
T12	Valparaiso	
T12	Washington	
14	San Diego State	
15	South Carolina	
16	Georgia	
17	Virginia Tech	
T18	BYU	
T18	Ohio State	
20	Monmouth	

Table 16: 2016 Tournament Selections

5 Conclusion

In this thesis, we explored two practical applications of BOSS. In the targeted marketing application, we observed that the BOSS method performed statistically significantly better than random selection with both raw and standardized data, and only very rarely performed worse than random selection. The performance also scaled linearly with the number of users in the control pool, which is a good sign for future use with larger datasets. Similarly, in the NCAA upset application, the BOSS method significantly outperformed weighted random selection and outperformed other state of the art models. These positive results indicate that BOSS could perform well in some real-world applications.

However, BOSS does have several limitations. The primary and perhaps most obvious limitation is that BOSS operates solely on groups. In a case where user-level information is required, the group nature of BOSS is not the best fit. For example, in the targeted marketing case, it is not possible with the current formulation to predict the rating of a given movie by a given user. Likewise, in the basketball upsets case, it is not possible to predict the probability of an individual game being an upset. An additional limitation is that BOSS optimizes the objective function defined by the user. Therefore, the choice of objective function can have a strong impact on the results. In this thesis we used two different objective functions, but there are many more that could be used, and those should be explored in the future in order to determine which are effective for which use cases. Furthermore, for larger datasets where solving the MIP is necessary, a linear objective function is required for satisfactory performance. This can cause problems if a linear function is not expressive enough for the desired result. Finally, performance could be in issue in some domains. While the performance is good enough to be used in many cases such as the ones tested, BOSS as presented does not seem suitable for real-time applications unless the dataset is extremely small. BOSS seems more suited to problems where solutions are required in minutes rather than in fractions of a second.

The positive results of BOSS applied to the targeted marketed and basketball upset cases suggests that BOSS, when used appropriately, has potential as a selection method in many applications across domains. While there are limitations, the performance demonstrated in the aforementioned cases is a strong indicator that BOSS can be applied to various problems, especially in conjunction with other standard machine learning techniques. Some future work exploring different potential objective functions and covariate selection methods would also be extremely helpful when considering BOSS for use in a new application.

6 References

Bell, Robert M., Yehuda Koren, and Chris Volinsky (2007). *The Bellkor solution to the netflix prize*. (Visited on 02/08/2015).

Breiman, Leo et al. (1984). "Classification and regression trees. Wadsworth". In: *Belmont, CA*.

Brenner, Jordan and Peter Keating (2015). *Upsets! We Got Upsets!* `http://espn.go.com/espn/feature/story/_/id/12348280/upsets-got-upsets`. (Visited on 08/30/2015).

Bryan, Kevin, Michael Steinke, and Nick Wilkins (2006). *Upset Special: Are March Madness Upsets Predictable?* SSRN Scholarly Paper ID 899702. Rochester, NY: Social Science Research Network. (Visited on 11/20/2015).

Candès, Emmanuel J. and Benjamin Recht (2009). "Exact Matrix Completion via Convex Optimization". en. In: *Foundations of Computational Mathematics* 9.6, pp. 717–772. ISSN: 1615-3375, 1615-3383. DOI: `10.1007/s10208-009-9045-5`. (Visited on 06/16/2015).

Cheung, Kwok-Wai et al. (2003). "Mining customer product ratings for personalized marketing". In: *Decision Support Systems*. Web Data Mining 35.2, pp. 231–243. ISSN: 0167-9236. DOI: `10.1016/S0167-9236(02)00108-2`. (Visited on 04/28/2016).

Covers (2015). *NCAA College Basktball Odds & Betting Lines – Spreads & Totals*. `http://www.covers.com/odds/basketball/college-basketball-odds.aspx`. (Visited on 10/21/2015).

ESPN (2015). *ESPN: The Worldwide Leader in Sports*. `http://espn.go.com/`. (Visited on 10/21/2015).

Ezekowitz, John (2013). *The RPI is Not the Real Predictive Indicator — The Harvard Sports Analysis Collective*. `https://harvardsportsanalysis.wordpress.com/2013/02/19/rpi-prediction/`. (Visited on 06/16/2016).

FiveThirtyEight (2015). *2015 March Madness Predictions — FiveThirtyEight*. `http://fivethirtyeight.com/interactives/march-madness-predictions-2015/\#mens`. (Visited on 10/21/2015).

Funk, Simon (2006). *Netflix Update: Try This at Home*. `http://sifter.org/~simon/journal/20061211.html`. (Visited on 03/28/2016).

Geurts, Pierre, Damien Ernst, and Louis Wehenkel (2006). "Extremely randomized trees". en. In: *Machine Learning* 63.1, pp. 3–42. ISSN: 0885-6125, 1573-0565. DOI: `10.1007/s10994-006-6226-1`. (Visited on 09/01/2015).

Goldberg, David et al. (1992). "Using Collaborative Filtering to Weave an Information Tapestry". In: *Commun. ACM* 35.12, pp. 61–70. ISSN: 0001-0782. DOI: `10.1145/138859.138867`. (Visited on 06/16/2015).

Kaggle (2015). *March Machine Learning Mania 2015 — Kaggle*. `https://www.kaggle.com/c/march-machine-learning-mania-2015`. (Visited on 12/22/2015).

Katz, Andy (2016). *ESPN.com - Memphis would have bumped Tulsa with win, chairman says.* http://espn.go.com/espn/print?id=14966925. (Visited on 06/16/2016).

Keating, Peter (2013). *Explaining the new Giant Killers formula - College Basketball.* http://espn.go.com/mens-college-basketball/story/_/id/9022008. (Visited on 08/30/2015).

Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix Factorization Techniques for Recommender Systems". In: *Computer* 42.8, pp. 30–37. ISSN: 0018-9162.

Lee, Wei-Po, Chih-Hung Liu, and Cheng-Che Lu (2002). "Intelligent agent-based systems for personalized recommendations in Internet commerce". In: *Expert Systems with Applications* 22.4, pp. 275–284. ISSN: 0957-4174. DOI: 10.1016/S0957-4174(02)00015-5. (Visited on 04/29/2016).

Lopez, Michael J.—Matthews (2015). "Building an NCAA men's basketball predictive model and quantifying its success". In: *Journal of quantitative analysis in sports* 11.1, p. 5. ISSN: 2194-6388.

Lü, Linyuan et al. (2012). "Recommender systems". In: *Physics Reports*. Recommender Systems 519.1, pp. 1–49. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2012.02.006. (Visited on 11/21/2014).

Marino, Jonathan (2015). *NCAA tournament gambling projection - Business Insider.* http://www.business-insider.com/ncaa-tournament-gambling-projection-2015-3. (Visited on 10/21/2015).

NCAA (2016). *March Madness bracket: How the 68 teams are selected for the Division I Men's Basketball Tournament — NCAA.com.* http://www.ncaa.com/news/basketball-men/article/2016-03-13/march-madness-bracket-how-68-teams-are-selected-division-i. (Visited on 06/16/2016).

Nikolaev, Alexander G. et al. (2013). "Balance Optimization Subset Selection (BOSS): An Alternative Approach for Causal Inference with Observational Data". en. In: *Operations Research* 61.2, pp. 398–412. ISSN: 0030-364X, 1526-5463. DOI: 10.1287/opre.1120.1118. (Visited on 11/03/2014).

Oliver, Dean (2013). *BPI -The College Basketball Power Index explained.* http://espn.go.com/mens-college-basketball/story/_/id/7561413/bpi-college-basketball-power-index-explained. (Visited on 08/29/2015).

Opitz, David and Richard Maclin (1999). "Popular Ensemble Methods: An Empirical Study". In: *Journal of Artificial Intelligence Research* 11, pp. 169–198.

Pomeroy, Ken (2011). *Basketball RPI: Why it's a lousy way to pick teams for the NCAA Tournament.* http://www.slate.com/articles/sports/sports_nut/2011/03/ratings_madness.html. (Visited on 06/16/2016).

Pomeroy, Ken (2012). *the kenpom.com blog.* http://kenpom.com/blog/index.php/weblog/entry/ratings_glossary. (Visited on 08/29/2015).

Sagarin, Jeff (2015). *College Basketball Ratings Page.* http://sagarin.com/sports/cbsend.htm. (Visited on 10/21/2015).

Sauppe, Jason J., Sheldon H. Jacobson, and Edward C. Sewell (2014). "Complexity and Approximation Results for the Balance Optimization Subset Selection Model for Causal Inference in Observational Studies". In: *INFORMS Journal on Computing* 26.3, pp. 547–566. ISSN: 1091-9856. DOI: 10.1287/ijoc.2013.0583. (Visited on 11/03/2014).

Silver, Nate (2015). *How FiveThirtyEight's March Madness Bracket Works*. (Visited on 08/30/2015).

Stephens, Matt (2015). *NCAA tournament makes example of Colorado State by saying RPI no longer important*. http://www.coloradoan.com/story/sports/csu/mens-basketball/2015/03/15/colorado-state-ncaa-tournament-rpi/24829067/. (Visited on 06/16/2016).

TeamRankings (2015). *Sports Predictions, Rankings & Stats - TeamRankings*. https://www.teamrankings.com/. (Visited on 08/30/2015).

Volner, Derek (2013). *Ride ESPN's BPI to bracket-making success - ESPN Front Row*. http://www.espnfrontrow.com/2014/03/bpi/. (Visited on 08/29/2015).

Wartenberg, Steve (2015). *How many people will fill out March Madness brackets? — The Columbus Dispatch*. http://www.dispatch.com/content/blogs/the-bottom-line/2015/03/how-many-people-will-fill-out-march-madness-brackets.html. (Visited on 11/03/2015).

Xu, Dongkuan and Yingjie Tian (2015). "A Comprehensive Survey of Clustering Algorithms". en. In: *Annals of Data Science* 2.2, pp. 165–193. ISSN: 2198-5804, 2198-5812. DOI: 10.1007/s40745-015-0040-1. (Visited on 04/26/2016).

Yahoo (2014). *$1 billion offered for perfect tournament bracket - Yahoo Sports*. http://sports.yahoo.com/news/1-billion-offered-perfect-tournament-200547143--ncaab.html. (Visited on 12/24/2015).

Yahoo (2015). *Yahoo Sports – Sports News, Scores, Rumors, Fantasy Games, and more*. http://sports.yahoo.com/. (Visited on 10/21/2015).

Appendix

A.1 Definitions of Basketball Statistics Used

Statistics and their definitions from TeamRankings.com (TeamRankings, 2015).

- Possession: One instance of a team controlling the ball until it scores, loses the ball, or commits a violation
- Steal: One instance of a defensive player forcing a turnover by acquiring or deflecting the ball from an offensive player
- Assist: One instance of a player passing the ball to a teammate in a way that directly leads to a field goal
- Effective Possession Ratio: (Possessions + Offensive Rebounds - Turnovers) / Possessions
- Games Played: Number of games a team has played in the current season before the tournament begins
- Extra Scoring Chances per Game: Offensive Rebounds + Opponent Turnovers - Opponent Offensive Rebounds - Turnovers
- Opponent Floor Percentage: Fraction of the opponent team's possessions that result in at least one point.
- Personal Fouls per Possession: Fouls / Possessions
- Opponent Steals per Defensive Play: Opponent Steals / Opponent Defensive Plays
- Assist / Turnover Ratio: Assists / Turnovers
- Personal Fouls per Defensive Play: Personal Fouls / Defensive Plays
- Opponent Steals per Possession: Opponent Steals / Opponent Possessions
- Opponent Average Scoring Margin: Average number of points between the opponent team and other teams they have played against (where positive is a victory and negative is a loss)
- Average Scoring Margin: Average number of points between the team and their opponents (where positive is a victory and negative is a loss)
- Opponent Three Point Percentage: Three Pointers made / Three Pointers attempted
- Steals per Defensive Play: Steals / Defensive Plays

- Steals per Possession: Steals / Possessions
- Average 2nd Half Margin: Average difference between the number of points the team scores in the 2nd half and the number of points their opponents score in the 2nd half.

A.2 NCAA Upset MIP Formulation

Let the following terms be defined:

- T: Treatment Group
- C: Control Pool
- G: Control Group
- $\mathbf{X}$: Set of covariates
- $\mathbf{X}$: Set of covariates
- T_i: Set of unique values in T for covariate $i \in \mathbf{X}$
- C_i: Set of unique values in C for covariate $i \in \mathbf{X}$
- V_i: Set of unique values in $T \cup C$ for covariate $i \in \mathbf{X}$
- $V_{i,j}$: j^{th} smallest value in V_i
- K_i: $K_i = K(T_i, G, V_i)$ for $i \in \mathbf{X}$: Kolmogorov-Smirnov statistic of the Treatment and Control groups for covariate $i \in \mathbf{X}$:
- R_i: $R_i = R(T_i, G)$ for $i \in \mathbf{X}$: Relative difference in sum between the Treatment and Control group for covariate $i \in \mathbf{X}$
- y_i: $y_i = \max(K_i, R_i)$: Larger of the Kolmogorov-Smirnov statistic and the relative difference between the treatment and control group for covariate $i \in \mathbf{X}$
- α_c: Binary variable which is 1 (0) if game $c \in C$ is (not) included in the control group, otherwise 0. The games with value 1 make up G.
- $x_{t,i}$: Value of covariate $i \in \mathbf{X}$ for $t \in T$
- $x_{c,i}$: Value of covariate $i \in \mathbf{X}$ for $c \in C$.

- $z_{i,j}$: $z_{i,j} = |\{x_{c,i} : x_{c,i} < V_{i,j}, c \in C\}|$: Number of units in control pool with value less than the jth smallest value of V_i for each covariate $i \in \mathbf{X}$
- $T_{i,j}$: $T_{i,j} = |\{x_{t,i} : x_{t,i} < V_{i,j}, t \in T\}|$: Number of units in the treatment group with value less than the jth smallest value of V_i for each covariate $i \in \mathbf{X}$
- β: $\beta = \sum_{c \in C} \alpha_c / |T|$: Constant that relates the size of the treatment group to the control group

Then, BOSS an be formulated as a Mixed Integer Linear Program (MILP)

$$\min \sum_{i \in \mathbf{X}} y_i \tag{37a}$$

such that

$$z_1 = \sum_{\substack{c \in C \\ \text{such that } x_{c,i} = V_{i,1}}} \alpha_c \tag{37b}$$

$$z_{i,j-1} + \sum_{\substack{c \in C \\ \text{such that } x_{c,i} = V_{i,j}}} \alpha_c = z_{i,j} \; \forall i \in \mathbf{X}, j \in \{2, 3, ..., |V_i|\} \tag{37c}$$

$$\frac{z_{i,j}}{\beta |T|} - \frac{T_{i,j}}{|T|} \leq y_i \text{ for all } i \in X, j \in \{1, 2, ..., |V_i|\} \tag{37d}$$

$$\frac{T_{i,j}}{|T|} - \frac{z_{i,j}}{\beta |T|} \leq y_i \text{ for all } i \in X, j \in \{1, 2, ..., |V_i|\} \tag{37e}$$

$$\frac{\sum_{c \in C} x_{c,i} \alpha_c - \beta \sum_{t \in T} x_{t,i}}{\beta \sum_{t \in T} x_{t,i}} \leq y_i \quad \text{for all } i \in \mathbf{X} \tag{37f}$$

$$\frac{\beta \sum_{t \in T} x_{t,i} - \sum_{c \in C} x_{c,i} \alpha_c}{\beta \sum_{t \in T} x_{t,i}} \leq y_i \quad \text{for all } i \in \mathbf{X} \tag{37g}$$

$$\sum_{c \in C} \alpha_c = \beta |T| \tag{37h}$$

$$\alpha_c \in \{0, 1\} \quad \text{for all } c \in C \tag{37i}$$

Equation (37a) minimizes the sum of the maximum of the Kolmogorov-Smirnov (KS) statistic and the relative difference for each covariate. Constraint (37c) sets the value of the empirical distribution at each point, and constraints (37d) and (37e) set the difference in empirical distributions at each point to be less than or equal to the KS statistic for that covariate. Constraints (37f) and (37g) set the relative difference constraints. Since the goal was to have BOSS select three teams, β was chosen depending on the size of the treatment group such that the control group would consist of three teams.