I. Introduction

The NBA is the premier basketball league of the world. It is an enormous business, with an estimated revenue of nearly $5 billion in 2014. Teams within the NBA with more wins will gain popularity and increase income, beneficial to the league and its players. Thus, predicting the win percentage of an NBA team based on the previous performance their roster is quite valuable to general managers, coaches, players, fans, gamblers, and statisticians alike. Moreover, knowing which particular stat or feature is most influential to winning games is also desirable. Our project aims to do exactly that.

We built a learning algorithm that takes in a data set of team features, pre-processed from previous statistics of each player in the current season’s team roster, and outputs an estimate of the win percentage of that team during the regular season. The input of our algorithm is a team feature vector of the 30 different NBA for several seasons. Each team in each season is considered one input data point. Each input data point vector is computed from taking the stats of all the players in the team combined by averaging or accumulating. The stats of the players in the team are however taken from one season prior. For example, if we want to compute the feature vector for the Golden State Warriors during the 2014-2015 season, we will look at all the players in the 2014-2015 GSW team, then find the statistics of each of these players during the 2013-2014 season and pre-processed them to create the required feature vector. The win percentage output would be the win percentage of the current, or the 2014-2015 win percentage of the GSW in our example.

The learning algorithm model that we use is a second order polynomial fit regression model. Unlike linear regression where the feature vector is used as it is, all the elements of our input feature vector will first be square and both the original value and the squared value are used as the final input features within the input vector. For example, we have 31 input features in our vector (without squaring). Before inputting this into our learning model, we will square the features to create a 62 feature vector. This vector includes the original 31 features plus the squared of each of the 31 features.

\[
\text{Input Feature vector } x^{(i)} = [x_1, x_2, ..., x_{31}, x_1^2, x_2^2, ..., x_{31}^2]^T
\]

\[
\text{Output Win Percentage } = y^{(i)}
\]

For \( i = 1, ..., n \) with \( n = \text{dataset size} \)

II. Related Works

There have been many previous research papers on predicting NBA team wins. However, most of them tries to predict wins of each individual NBA games, then sum them up to then create the win percentage of a season. Two papers, one by Bernard, L et al. (2009)[1] and the other by Pedro et al. (2012)[2] used network based algorithm to predict NBA wins. Because mere statistical features do not produce as well of a result (as can be seen in our model results) a network based feature model can definitely improve learning algorithms. More specifically, Bernard, L et al. used neural networks with feature selection to predict NBA games. In our opinion, this is a very good model to use, since it takes in many sophisticated nature of many features (ones we weren’t able to incorporate into our model).

Another interesting model done by Na Wei (2011)[3] utilizes Naive Bayes to predict playoff records from regular season stats. Even though this is a different problem, the nature of the problem is similar. The use of Naive Bayes however is not a very good approach. Since the model relies on a lot of assumptions, the model is not as accurate.

Online, statistical resources for sports fans span an enormous spectrum. Basketball aficionados of all kinds and skills can enjoy interacting with the Buckets visualized shot chart from BBall Breakdown[4] as well as studying the complex decision making process of Markov state machines on Big League Insights[5] as applied to the game. Using data from thousands of basketball plays, the model analyzes and predicts how multiple in-game states and transitions influence each other and the overall outcome of a game.

III. Dataset and Features

A. Data Sources

We collected our data from two main sources. The first main source where we collected most player statistics during each seasons, as well as team statistics (team roster data of each seasons and team win percentage of each seasons), is the main NBA statistics website http://stats.nba.com/league/player/. The second source that we obtained our feature stats from, especially the team salaries stats is https://www.eskimo.com/pbender/.
B. Data set pre-processing

As described in the introduction, our input data is the set of team statistics for each season taken from player roster statistics from the season before. The raw data we obtained are therefore every player in the league during the 2004-2005 season to the 2013-2014 season. We then pre-processed this to obtain 30 team feature statistics for the year 2005-2006 to 2014-2015.

These features include:

1) Age 16) Rebounds (REB)
2) Games Played (GP) 17) Assists (AST)
3) Wins (W) 18) Turnovers (TOV)
4) Minutes (MIN) 19) Steals (STL)
5) Field goals Made (FGM) 20) Blocks (BLK)
6) Field goals Attempted (FGA) 21) Personal Fouls (PF)
7) Field goal % (FG%) 22) Double-double (DD2)
8) 3-points Made (3PM) 23) Triple-doubles (TP3)
9) 3-points attempted (3PA) 24) Points (PTS)
10) 3-points % (3P%) 25) Efficiency (+/-)
11) Free-throws Made (FTM) 26) # of rookies
12) Free-throws attempted (FTA) 27) # of allstars
13) Free-throws % (FT%) 28) # of Rebounders
14) Offensive rebounds (OREB) 29) # of Double-doubles
15) Defensive rebounds (DREB) 30) # of Triple-doubles

The Number of allstars is defined as the number of players with points per game of 1 standard deviation above the mean of a season. Similarly, number of rebounders is the number of players with rebounds higher than 1 standard deviation. Likewise with number of double-doubles and triple-doubles.

The 1st feature up to the 25th feature are obtained from averaging the stats of each player in the team per season together. The last 5 features are obtained from counting the players in the team per season with the feature requirements.

We then added the last or 31st feature, called team salaries, using the team’s available salaries during the current season (so from 2005-2006 to 2014-2015 seasons). All of the 31 features are then normalized per season by whitening. With a total of 10 seasons and 30 teams, we have a total of 300 data points in our data set.

C. Extra Data Manipulation

We tried to run PCA or Principle Component Analysis to remove any dependent features. A new feature vector, with reduced dimension/size, is created. The size of the new feature was determined by how many dominant eigenvalues of the empirical co-variance matrix \( \Sigma = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} x^{(i)T} \) there is (where \( m = \) the number of train samples or data points and \( x^{(i)} \) is the input feature vector of data point \( i \)).

Suppose there are \( k \) dominant eigenvalues (\( k < 31 \)), then the new input feature will be:

\[
\begin{bmatrix}
    u_1^T x^{(i)} \\
    u_2^T x^{(i)} \\
    \vdots \\
    u_k^T x^{(i)}
\end{bmatrix}
\]

Where \( u_j \) are the eigen-vector corresponding to the \( j^{th} \) eigenvalue of \( \Sigma \).

However, when the learning model is run using the reduced feature vector from PCA versus the original feature vector, there was no improvements (the generalized error even went up) and so we decided not to use the reduced vector from PCA. Instead, we decided to reduce the feature vector using feature selection, more specifically backward search. This method proves to give us a better result. We will describe how we did feature selection in more detail in the methods section.

IV. Method

The learning algorithm that we chose to use for our learning model is the second order polynomial fit regression model. However, before we decided on this model, we ran our data through three different regression models, linear regression, locally weighted linear regression and polynomial fit (with different orders). We validated each method/model using k-fold validation and chose the model that gave the best generalization error (lowest RMS error of the test set).

For each regression model, we measured their performances by measuring the cost function \( J(\theta) \) of both the training set and the test set. The cost function we used is defined as follows:

\[
J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)})^2
\]

For locally weighted regression, we varied the parameter \( \tau \) between \( \tau = 10 \) and \( \tau = 40 \). For polynomial fit, we ran our data through second order and third order polynomial fits.

Our data-set includes 300 data points, 30 teams during 10 seasons. We decided to validate our learning algorithm by separating this data-set into the training set and the test set. We separated 270 data samples as the training set and the other 30 as the test set. For simplicity we used one season as the 30 data test set and the other seasons as data to train. Because of the nature of how we separated the data and because there are 10 seasons, we decided to use 10-fold validation. Therefore, each season becomes a test set once. The pseudo-code for this 10-fold validation then becomes:

1) Split the data into 10 disjoint sets, \( S_1, S_2, ..., S_{10} \), where each set contains data from each of the 10 seasons.
2) For \( j = 1, ..., 10 \):
   - Train the model \( M \) on \( S_1 \cup ... \cup S_{j-1} \cup S_{j+1} \cup ... \cup S_{10} \) (train the model on every set except set \( S_j \)) to obtain the hypothesis \( h_j \). Then test the hypothesis on \( S_j \) to get the generalization error \( \epsilon_j \).
3) The generalization error $\hat{\epsilon}$ for model M is then the average of all the $\epsilon_j$ for $j = 1, ..., 10.$

10-fold validation is done with every regression model for comparison.

A. Regression Model Descriptions

In this section, we will briefly describe how each regression model, linear regression, locally weighted linear regression, and polynomial fit regression, works.

We will start with linear regression. In linear regression, we are using the data to create a linear model in the form $y = h(x) = \theta^T x$ where $\theta = \theta_0 + \theta_1 x_1 + ... + \theta_{31} x_{31}$ with $x \in \mathbb{R}^{31}$. The model aims to minimize the objective function $\frac{1}{2} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2$.

To minimize the object function, we used the normal equation defined by:

$$\theta = (X^T X)^{-1} X^T \bar{y}$$

$$X = \begin{bmatrix}
    (x^{(1)})^T \\
    (x^{(2)})^T \\
    \vdots \\
    (x^{(31)})^T \\
\end{bmatrix}$$

$$\bar{y} = \begin{bmatrix}
    y^{(1)} \\
    y^{(2)} \\
    \vdots \\
    y^{(31)} \\
\end{bmatrix}$$

Next for polynomial fit with different orders, we ran our data on second order and third order. For each polynomial fit regression, we are trying to minimize the same objective function as the case for linear regression. However, the only difference is in the hypothesis $h(x)$. For second order polynomial fit, we have $h(x) = \theta^T \hat{x} = \theta_0 + \theta_1 x_1 + ... + \theta_{31} x_{31} + \theta_{32} x_1^2 + ... + \theta_{62} x_{31}^2$. For this case, $\theta \in \mathbb{R}^{62}$ and $\hat{x} \in \mathbb{R}^{62}$. For the case of a third polynomial fit, we have $h(x) = \theta^T \hat{x} = \theta_0 + \theta_1 x_1 + ... + \theta_{31} x_{31} + \theta_{32} x_1^2 + ... + \theta_{62} x_{31}^2 + \theta_{63} x_1^3 + ... + \theta_{93} x_{31}^3$. For this case, $\theta \in \mathbb{R}^{93}$ and $\hat{x} \in \mathbb{R}^{93}$.

Lastly, for locally weighted linear regression, we try to fit $\theta$ to our model to minimize the objective function, $\sum_i w^{(i)} (y^{(i)} - \theta x^{(i)})^2$. To do this we also used a modified normal equation in the form:

$$\theta = (X^T W X)^{-1} X^T W \bar{y}$$

with $W_{ii} = \frac{1}{2} w^{(i)}$, $W_{ij} = 0$ for $i \neq j$

B. Learning Model Improvements

After we decided on second order polynomial fit regression, we worked on improving our learning model by trying to reduce our variance using feature selection. The type of feature selection we chose was backward search. To do this we ran the following algorithm:

1) Initialize $\mathcal{F} = \{1, ..., 31\}$ with our 31 features.

2) Repeat { 
(a) For $i = 1, ..., 31$, delete feature $i$ from the set $\mathcal{F}$ and call it the set $\mathcal{F}_i$. Then use cross-validation to evaluate the generalization error of the new $\mathcal{F}_i$. 
(b) Set $\mathcal{F}$ to be the set $\mathcal{F}_i$ in (a) with the least generalization error. 
}

3) End the repeat loop when the generalization error does not improve any longer or if it starts to increase again with fewer features.

V. Results & Discussions

We will split our results discussion into three parts. The first part is the results obtained while choosing the best regression learning algorithm between the ones we have mentioned. The second part will be the results obtained from trying to improve the model using different techniques like increasing the data set size and feature selection (also PCA even though it did not improve the model). The third part will be the end result or the final model we achieved and how we utilized our model on the current 2015-2016 NBA season.

A. Different Regression Model Results

As mentioned, we first learn our data on different regression models. We measured the cost function of each model using k-fold validation and decided on the model with the least RMS generalization error or $J(\theta) = \sqrt{\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2}$. Initially, we were working with data from 6 seasons instead of 10 seasons, from 2009-2010 to 2014-2015. With this dataset we validated each model using 6-fold validation.

We first ran our dataset with linear regression. We obtained a training RMS error of 18.9 and a test RMS error (or generalization error) of around 20. The result gives a good variance but a very bad bias problem. We therefore decided to fit our data using different polynomial fits. Second order fit gave us better RMS training data error of 11.4 and third order gave us an even better RMS training data error of 8.2. However, both second order and third order fits produced worse RMS test error of 22.7 and 29.3 respectively. Therefore, we improved our bias problem with the higher order fit but created a high variance problem (an over fitting problem).

We tackled our high order fit’s variance problem by increasing the dataset to 10 seasons. This reduced the RMS test error of all the polynomial fit orders (including first order or linear regression) down. The resulting RMS test error of linear regression, second order fit, and third order fit are 17.4, 16.9, and 18.0 respectively. Although the RMS test error decreased, the RMS training error increased for all three models with errors of 15.9, 13.5, and 12.1 respectively. Even if the training error increased, the test error decreased which means that the
generalization error improved and so increasing the dataset did make a good improvement. A plot showing this result is shown below.

We then varied the dataset size from 6 seasons to 10 seasons for second order and first order polynomial fit and plotted a graph to see the difference. The graph is shown below.

The RMS test error or generalization error did not fall below 17.0 with the \( \tau \) varied and so the model with the best RMS test error (the lowest error) is therefore second order polynomial fit.

B. Model Improvements - Feature Selection

With second order polynomial fit model, we have a RMS training error of 13.5 and RMS test error of 16.9. We have already mentioned how we improved our test error and variance problem by increasing the dataset size to 10 NBA seasons from 6 NBA seasons (or from 180 data samples to 300 data samples).

We further improved our generalization error by doing feature selection using backward search. We managed to reduce our feature vector of 31 features down to 14 features. This means that our input vector would now have a size of 28 instead of 62 for second order polynomial fit. The size of the feature vector reduced by more than half. The 17 features that were reduced are shown below in order of removal:

1. num_rookies
2. PF
3. team_salaries
4. TOV
5. DREB
6. FGA
7. FG%
8. FT%
9. FTA
10. 3P%
11. PTS
12. MIN
13. FTM
14. AGE
15. REB
16. OREB
17. BLK
Fig. 4. Plot showing the RMS test and train error changes as features are removed

With the features removed, we managed to reduce the RMS test error or generalization error down to 15.54, with a training error of 14.18.

C. End Results

Our resulting hypothesis \( h(x) = \theta^T x \) with \( x, \theta \in \mathbb{R}^{28} \), with all our improvements computed, has \( \theta \) corresponding to the first order and second order weights of the feature as follows:

<table>
<thead>
<tr>
<th>Features Used ((x_1 \text{ to } x_{14}))</th>
<th>First Order Weights to ( \theta_{14} )</th>
<th>Second order Weights to ( \theta_{28} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>0.016</td>
<td>-0.114</td>
</tr>
<tr>
<td>W</td>
<td>1.893</td>
<td>-0.031</td>
</tr>
<tr>
<td>FGM</td>
<td>-2.873</td>
<td>0.138</td>
</tr>
<tr>
<td>3PM</td>
<td>3.758</td>
<td>0.192</td>
</tr>
<tr>
<td>3PA</td>
<td>-4.212</td>
<td>-0.047</td>
</tr>
<tr>
<td>AST</td>
<td>2.288</td>
<td>-0.006</td>
</tr>
<tr>
<td>STL</td>
<td>1.078</td>
<td>0.035</td>
</tr>
<tr>
<td>DD2</td>
<td>-2.830</td>
<td>0.178</td>
</tr>
<tr>
<td>TD3</td>
<td>-1.705</td>
<td>0.095</td>
</tr>
<tr>
<td>+</td>
<td>-1.141</td>
<td>0.004</td>
</tr>
<tr>
<td>Allstars</td>
<td>15.754</td>
<td>-2.412</td>
</tr>
<tr>
<td>Rebounders</td>
<td>6.977</td>
<td>-0.881</td>
</tr>
<tr>
<td>num_double doubles</td>
<td>11.528</td>
<td>-1.731</td>
</tr>
<tr>
<td>num_triple doubles</td>
<td>6.725</td>
<td>-0.821</td>
</tr>
</tbody>
</table>

As seen in the table of the values of \( \theta \) obtained from our model, the feature that is most influential on win percentage of NBA teams in a season is the number of all-stars. This feature is the number of players with a point per game of 1 standard deviation above the average. It has the largest corresponding \( \theta \) weight of 15.754. The next most influential feature is the number of players with double-doubles of 1 standard deviation more than the mean in a season. The corresponding \( \theta \) weight of this feature is 11.528.

With our model, we predicted the win percentage of current NBA regular season teams as follows:

Fig. 5. Plot showing our win percentage predictions of NBA teams during the 2015-2016 season

The season is still ongoing and thus the accuracy and errors cannot be determined yet.

VI. CONCLUSION & FUTURE WORKS

With our chosen learning algorithm/model and tools to improve our model, such as feature selection, we got our generalization RMS error down to 15.54. This is a big improvement from an error of more than 20 in the beginning. The second order polynomial fit is the best model we found because linear regression seems to under fit the data, whereas higher order polynomial fits (i.e. third order or higher) over fits the data.

With more time and computational resources, we believe that the error could be reduced further. To do this we would need to gather more detailed data such as different plays run or individual team head-to-head games and schedule to create more relevant and detailed features. Some examples of these features are coach stats, different plays run by team coaches and their frequencies, etc. These data are hard to get and hard to pre-process before being able to add to the model. Also with better knowledge on other learning algorithms not taught in class, we believe that using a neural network algorithm approach would give a better result overall.
VII. REFERENCES


