<https://www.overleaf.com/9987293889qhpggyhdbjhc>

Highlight means complete

1. **Introduction (0.5-1pgs)**
2. **Problem Statement and Dataset (0.5 -1pgs)**

* Problem Statement: Gambling is good but hard (only place big bets, duh) (big bets = big rewards)
* Datasets:
  + Mention datasets we considered and how we narrowed it down (years covered, and extra special features )

1. **Methods and Models (1-2pgs)**

How we did:

* Data Engineering
  + Implementing rolling average for 5, 10, 15 games in both datasets
* Feature Selection
  + Determine which features if any are correlated with the output
  + This will indicate which features are most important with making a prediction
  + irrelevant features can then be removed
  + Side effect: faster model training
  + Numerical inputs, Categorical outputs
  + ANOVA
* Data Normalization
  + Normalized to be between -1 and 1
  + This is important for Logistic Regression, SVMs, and NNs
  + This is from a Lecun paper I can find if need - Reggie
* Model Selection
  + Models were selected based on what we found in literature
* Model Implementation
  + Sklearn for LR, SVM, Decision Trees
    - Include parameters such as epochs, tolerances, learning rates, etc
  + Keras/Tensorflow for NNs
  + Hyperparameter tuning
    - Optuna
    - Split the data into training, testing, and validation
    - Train model on training, choose models based on validation, results are from testing set
* Bias variance analysis
  + Quick description of what is done for this

1. **Results and Discussions (1-2pgs)**

Bias Variance Tradeoff Analysis -

* 1. Compare the different averages + stdevs?
  2. Say they’re pretty similar? I’m assuming that we’re all really close with our averages and stdevs (no overfitting?) - maybe need more features since we’re not that accurate? - J
  3. According to the outline we need to show how we used bias-variance to guide our next steps so I think we need to talk about how we needed more complex models than log reg due to the bias error I assume it has
  4. Then we talk about our individual models what we see, I haven’t run my non regularization yet but I hope it has variance so I can justify adding regularization LOL. And for James you ran different kernels right? yes <- great so you can mention you tried that to accommodate the high bias error. I think this works for decision trees -> random forests as well
  5. We’ll use log reg for our baseline? - then talk about going to our more complex models? Yeah
  6. Okay, i’ll quickly write something up so you can compare it to that
  7. Great

Model Regularization

* 1. Log regression -> L1, L2, elastic net
  2. Decision Trees -> Random Forests
  3. SVM -> tried different kernels
  4. Gradient Boosting -> turned regularization parameters on and off
  5. NNs -> applied L1 and L2 regularization
  6. Decision Trees, SVM, didn’t have regularization in the same sense as NNs
     1. Applied changes to the base models to see if that would improve performance
  7. Results will be discussed in further detail in later section

1. **Implementation and Code (0.5 pgs)**

-pandas for data formatting

-scikit-learn for the different models and data splitting

-optuna for hyperparameter optimization

1. **References**

-copy from proposal

1. **List of figures that we want to add**
   1. Final model accuracies (for results)
   2. Model parameters (for results)
   3. ANOVA results (for methods)

**Word Vomit - to be edited and put into final report below**

**Proposal (Edited version in final report)**

Using Machine Learning (ML) for predictive sports modeling is a very well documented field, with basketball being the second most studied sport [1]. It has been found that basketball games are homogeneous processes up to the beginning end of each quarter [2]. Our intended models were found to be some of the most popular ML models for sports prediction,with artificial neural networks being the most selected by far (and support-vector machines (SVM) performing best in many instances, e.g. [3] [4]). It has been shown that this type of ML can be used to generate a profit from sports betting [5], however, results are still relatively inaccurate,where it has been shown that there is a glass ceiling of 75% accuracy for predictive sports modeling [6]. This might be due to the fact that although researchers may vary their models, similar features are being selected. Also, data sets vary drastically from study to study, which inhibits researchers from constructively comparing their findings with previous work, leading to unclear development [7]. Feature selection has been argued to be more important than model selection,so we will make sure to do our due diligence in selecting the optimal features for this project. We can look into player-level statistics, or use play-by-play data to select features, as this adds context to the general team and box-score data [4].B. Data CollectionData can be collected from a variety of sources. There are multiple Kaggle datasets that will fit the needs of this project.If additional data is needed then resources such as NBA.com,or basketball-reference.com may be used. Initially a data set may need to be put together manually by combining multiple sources.There are a few different types of datasets that may be used. Initially we will be examining the rolling average of the last 5 games statistics. This 5 game value will also be varied to 10, and 15 games. The statistics that we average also have some variations, there are some advanced statistics we can use (for example: team offensive rating, defensive rating), orsome basic statistics (for example: field goal percentage, turnovers,rebounds, assists, etc). These statistics will be included in the initial data, then some feature selection will be used to evaluate the prediction ability.Once these statistics are finalized, data normalization will be used. This is especially important for SVM’s & ANNs, but not Decision Trees or Gradient Boosting.C. Feature SelectionThere are several feature selection algorithms that we can use. Firstly we will be using ANOVA testing to determine the relationship between the features of the data set and thetarget. ANOVA testing will be used because the input data is numerical and the output will be binary classification, where0 equals home team wins and 1 equals away team wins. From This testing we may choose a smaller subset of features to use,but we won’t know until we do this testing.D. Machine Learning ModelsAfter finding some related work, we have narrowed down the models that we plan to use to the following:•Gradient Boosting (XGBoost)•Neural Networks•SVM•Decision TreesFor now, the parameters of these models will be optimized using Grid Search. If a better library or method is found before this step, it may be used

**Data Acquisition (should be split into “Problem Statement and Dataset” and “Data Engineering”**

Professional sports leagues, especially the NBA, are very well documented when it comes to data collection. Websites such as the official NBA website, Basketball reference, and ESPN, have team data, box-score data, and player-level data for games dating back to the middle of the 20th century. However, the late 1980s are when we noticed a significant improvement in data collection, where the data is much more comprehensive and in-depth (we started seeing detailed player level statistics, advanced performance metrics, etc.).

There are many ways the data can be extracted, and there is a vast amount of comprehensive datasets available for us to use online. We narrowed our research down to the NBA Enhanced Box Scores and Standings (2012-2018) database from Kaggle ([https://www.kaggle.com/pablote/nba-enhanced-statistics](https://www.kaggle.com/pablote/nba-enhanced-stats)), which includes data for every regular season game starting from the 2012-2013 season, up until the 2017-2018 season. This dataset was created on Java using RESTful APIs provided by xmlstatistics (https://erikberg.com/api). It is a comprehensive dataset, including player-level statistics for each game, down to the quarter. It also includes detailed team-level statistics, which tally the player-level data for each game, and calculaute advanced statistics such as Efficiency Differential, and Offensive/Defensive ratings which proved to be valuable (will be discussed in more detail in the feature selection section). These statistics are calculated based on formulas found on websites such as NBAStuffer, Basketball-Reference, and AdvancedProBasketball, and we used them to recalculate possession and pace for each game.

(<https://www.nbastuffer.com/analytics101/>

<https://www.basketball-reference.com/about/glossary.html>

<https://advanceprobasketball.com/sozluk/>)

Basketball is an evolving sport, where teams and players are constantly changing the dynamics of the league. The 2012-2018 period is representative of the dynamics of today’s NBA as it includes the majority of today’s most dominant teams and players, and includes at least one cycle of the most recent shift in these dynamics, which is the threshold of what our model might be sensitive to (instead of multiple cycles and rebuilds over time, which might need a different approach, and will be discussed in the future work section). We have over 7500 games to train our models on, which was sufficient and produced results that are in-line with our literature review.

The objective of this project was to use previous performances to predict the success of a matchup. The team box-score data is laid out where there are two rows per game, one row for the home team, and one row for the away team. Each row had two sets of columns with team statistics and opponent statistics (as in the statistics for each game was repeated twice from different perspectives, which is useful for some sports-modeling).

First, the data was sorted based on game date and the team IDs. A dictionary was created that is indexed by team ID and game date to store the raw data from the dataset. An additional key was added for the win percentage (at this point, before averaging this consisted of ones and zeroes for wins and losses, respectively). Irrelevant columns, such as team division, and referee names were disregarded in our model.

To acquire a statistics average with the last n games, this ordered dictionary was required. The statistics were averaged for the last 5, 10, and 15 games in separate datasets. For each team ID, an average was taken at each game date consisting of the previous n games; this data replaced the dictionary row containing the raw data – the only parts of the data untouched were the win or loss column (to be used for the validation labels, which our algorithm generated). These averaged data points then replaced the original non-averaged row in the initial data frame where the data was first read. This resulting dataset was then in a form that we didn’t want to work with. In this form, the data read such that each game had two rows: one for the home team with the statistics of the previous opponents that they had faced, and one for the away team of a game with the previous statistics of the opponents they had played.

We tackled this by going into the original dataset and deleting the opponent columns entirely. This left us with only one column of data, that had a row of home team data and away team data for each game. Running our original averaging algorithm on this version of the dataset meant that for each row, we now have an average of how that specific team performed over the last n games. We then sorted the dataset manually by “Home”/”Away”, followed by sorting each category by game date, followed by concatenating both blocks in a way that left us with one row per game – a set of columns with home statistics, and a set of columns with away statistics (this is why we had to recalculate pace and possession). We also came up with an alternative algorithm, where the original program was amended to average home and away performances for each team exclusively, but this version of the dataset proved to be redundant.

**FINAL REPORT**

**Introduction (0.5-1pg) (subheadings for organization, will be removed after)**

-Problem (Honestly didn’t know what to say so I made it cheesy, please review/edit)

Sports are a favourite pastime for people around the world. This global popularity has resulted in a huge number of statistics being collected and organized, a goldmine for data scientists. This makes predicting the outcome of sporting events both a fun pastime, and a lucrative one when it comes to sports betting. Using Machine Learning (ML) for predictive sports modeling is a very well documented field, with basketball being the second most studied sport [1].

-Significance and Challenge

It has been shown that ML can be used to generate a profit from sports betting [5]. In this case the greater the accuracy of the models the greater the potential profit. However, even with the amount of current documentation that exists, there appears to be a glass ceiling of 75% accuracy in current models [6]. This could stem from the fact that, even with the varying amount of models that have been used, similar features are being selected. Feature selection has been argued to be even more important than model selection in this problem. Another issue is that data sets vary drastically from study to study, inhibiting researchers from constructively comparing their findings with previous work and leading to unclear development [7].

-Prior Work (didn’t have much on past work in the proposal)

Datasets for basketball statistics are maintained by the NBA on their website, as well as multiple Kaggle datasets which fit the needs of this project. The most popular models for sports prediction tend to be artificial neural networks and support-vector machines, with the latter performing the best in many instances [3][4].

-Overview of Report

A dataset from something or another was selected for this project, due to the specific statistics it had. The statistics were averaged for each team over the last 5, 10, and 15 games. These averages were separated into three different datasets for comparison. ANOVA tests were conducted on the datasets to select the top 15 features. Once those features were extracted, the final dataset was normalized. Initially, logistic regression was used as a baseline model. More complex models were selected from the literature to be: Neural Networks, SVM, Decision Trees, and XGBoost. All models were tuned and iterated before a final test was run for comparison.

**Problem Statement**

There is plenty of compiled data made public by professional sports organizations such as the NBA. This data can be used to predict the outcomes of future matchups which can benefit parties such as sports analysts, ticket sales, fans, and those involved in sports betting. The goal of this study is to analyze the performance of different machine learning algorithms and select our optimum algorithm for predicting match outcomes.

**Dataset**

**\*\* updated \*\***

With the vast collection of extensive data in professional basketball found on websites such as the Official NBA website, Basketball Reference, and ESPN, we needed to compile data that was appropriate for the modern game. This data collection dated back to the middle of the 20th century with more detailed player level statistics, advanced performance metrics, etc. starting to appear in the 1980s. To predict team performance, we needed a dataset with these more detailed statistics over many seasons.

We narrowed our research down to the NBA Enhanced Box Scores and Standings (2012-2018) database from Kaggle ([https://www.kaggle.com/pablote/nba-enhanced-statistics](https://www.kaggle.com/pablote/nba-enhanced-stats)), which includes data for every regular-season game starting from the 2012-2013 season, up until the 2017-2018 season. This dataset was created on Java using RESTful APIs provided by xmlstatistics (https://erikberg.com/api). It includes player-level statistics for each game, down to the quarter. It also includes detailed team-level statistics, which tally the player-level data for each game, and calculaute advanced statistics such as Efficiency Differential, and Offensive/Defensive ratings which proved to be valuable (will be discussed in more detail in the feature selection section). Moreover, the 2012-2018 period is representative of the dynamics of today’s NBA as it includes the majority of today’s most dominant teams and players. We also had over 7500 games to train our models on, which was sufficient and produced results that are in-line with our literature review.

The team box-score data is laid out where there are two rows per game, one row for the home team, and one row for the away team. Each row had two sets of columns with team statistics and opponent statistics (as in the statistics for each game was repeated twice from different perspectives, which is useful for some sports-modeling).

**\*\* previous stuff \*\***

Professional basketball has extensive data collection resources. Websites such as the official NBA website, Basketball reference, and ESPN, have team data, box-score data, and player-level data for games dating back to the middle of the 20th century, with detailed player level statistics, advanced performance metrics, etc. starting to appear around the 1980s. The objective of this project was to use previous performances to predict the success of a matchup. In order to do this, we needed a dataset that documents team performance for every matchup, over many seasons.

We narrowed our research down to the NBA Enhanced Box Scores and Standings (2012-2018) database from Kaggle ([https://www.kaggle.com/pablote/nba-enhanced-statistics](https://www.kaggle.com/pablote/nba-enhanced-stats)), which includes data for every regular-season game starting from the 2012-2013 season, up until the 2017-2018 season. This dataset was created on Java using RESTful APIs provided by xmlstatistics (https://erikberg.com/api). It includes player-level statistics for each game, down to the quarter. It also includes detailed team-level statistics, which tally the player-level data for each game, and calculaute advanced statistics such as Efficiency Differential, and Offensive/Defensive ratings which proved to be valuable (will be discussed in more detail in the feature selection section). Moreover, the 2012-2018 period is representative of the dynamics of today’s NBA as it includes the majority of today’s most dominant teams and players. We also had over 7500 games to train our models on, which was sufficient and produced results that are in-line with our literature review.

The team box-score data is laid out where there are two rows per game, one row for the home team, and one row for the away team. Each row had two sets of columns with team statistics and opponent statistics (as in the statistics for each game was repeated twice from different perspectives, which is useful for some sports-modeling).

**Methods and Models (1-2pgs)**

**Data Engineering**

We needed to get the data in a format where every row presents the rolling average of statistics for either team in the match-up over the past n games. This smooths the data and removes noise for implementing our classification models. First, the data was sorted based on game date and the team IDs. A dictionary was created that was indexed by team ID and game date to store the raw data from the dataset. An additional key was added for the win percentage (at this point, before averaging this consisted of ones and zeroes for wins and losses, respectively). Irrelevant columns, such as team division, and referee names were disregarded in our model.

To acquire a statistics average for the last n games, this ordered dictionary was required. The statistics were averaged for the last 5, 10, and 15 games in separate datasets. For each team ID, an average was taken at each game date consisting of the previous n games; this data replaced the dictionary row containing the raw data – the only parts of the data untouched were the win or loss column (to be used for the validation labels, which our algorithm generated). These averaged data points then replaced the original non-averaged row in the initial data frame where the data was first read. This resulting dataset was then in a form that we didn’t want to work with. In this form, the data read such that each game had two rows: one for the home team with the statistics of the previous opponents that they had faced, and one for the away team of a game with the previous statistics of the opponents they had played.

We tackled this by going into the original dataset and deleting the opponent columns entirely. This left us with only one column of data, that had a row of home team data and away team data for each game. Running our original averaging algorithm on this version of the dataset meant that for each row, we now have an average of how that specific team performed over the last n games. We then sorted the dataset manually by “Home”/”Away”, followed by sorting each category by game date, followed by concatenating both blocks in a way that left us with one row per game – a set of columns with home statistics, and a set of columns with away statistics (this is why we had to recalculate pace and possession). We also came up with an alternative algorithm, where the original program was amended to average home and away performances for each team exclusively, but this version of the dataset proved to be redundant.

**Feature Selection**

This task was dependent on the features that are present in the dataset. In this case, we have a set of numerical input features with a binary categorical output. Because of this combination we used ANOVA testing to determine the correlation between each feature and the output. The feature selection algorithms from the package scikit-learn were used to perform this task. Once the correlation between features was visualized, the top 15 features were selected to be used as the inputs to our models. The number 15 was used after a visual inspection of the correlations, as 15 seemed to encompass all of the highly correlated features as well as a few less correlated features.

**Data Normalization**

Algorithms that rely on combining/manipulating features through feature mapping can run into issues when there is a large difference in the scale between features. This also presents a problem when updating weights through gradient descent as unscaled data can cause large errors for initial parameter guesses in the update rule resulting in an unstable learning process.

<https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/>

<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/normalize-data>

**Model Selection**

Models were selected based on what was found in relevant literature. ANNs and SVM have been extensively used in this domain previously \cite{3} \cite{4}. Decision trees have been applied to other applications in the sports domain, namely predicting the hall of fame status of Major League Baseball players. We applied decision trees to our problem because we wanted to see if it was efficient in another sports domain. Similarly to decision trees, gradient boosting was chosen because of its efficiency in predicting English Premier League Team Season Win Percentages \cite{}. We thought that this application was similar to our problem and wanted to include it for that reason.

**Model Implementation**

For this project, models were implemented from pre-existing libraries such as scikit-learn and Keras. For a baseline model, logistic regression was chosen due to the problem being binary classification. The more complex models that were implemented are: Decision Trees, XGBoost, SVM, and Neural Networks. For each model the dataset was split into a ratio of 70/10/20 for training/validation/test datasets. Optuna was used to tune the hyperparameters of each model, with the objective of maximizing the validation accuracy. This was done for each dataset (rolling 5, 10, and 15 game averages). Once a tuned set of hyperparameters was obtained for each situation, the models were trained on the training dataset and ran predictions on the validation set. This training/predicting process was run 10 times for each situation to provide statistical weight behind the results. The average training and validation accuracies were compared for bias-variance analysis to determine if regulation needed to be added/reduced or if a more complex model was needed. The model was then re-tuned with the added changes and the bias-variance analysis was done on the new results. Once a model was finalized it was trained on the training dataset and ran predictions on the test data set for the final results. This was also done 10 times and average for statistical significance. For the final comparison between models, precision and recall were compared in addition to accuracy.

**Results and Discussions (1-2pgs)**

Log Reg

The first model tested for this problem was a logistic regression model without regularization using up to 10000 epochs and a tolerance of 0.0001. The 15-game dataset using n=10 independent trails was the most accurate with an accuracy of 0.661 +/- 0.0021 on the training set, an accuracy of 0.665 +/- 0.014 on the validation set, and an accuracy of 0.651 +/- 0.0017 on the test set. As this was the simplest model, this served as a baseline for the more complex models. Based on the similarity between the validation set and the training set, there was very little variance error. However, at 65 percent accuracy, there was significant bias error.

To find improvements in this model, three types of regularization were applied – an L1 regularizer, an L2 regularizer, and an elastic-net regularizer (a combination of L1 and L2). The hyperparameters for these models were tuned, and the following accuracies were obtained for the 15-game dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Base model | L1 regularization | L2 regularization | Elastic-net |
| Training accuracy | 0.661 +/- 0.0021 | 0.662 +/- 0.0022 | 0.662 +/- 0.0015 | 0.662 +/- 0.0016 |
| Validation accuracy | 0.665 +/- 0.014 | 0.654 +/- 0.019 | 0.660 +/- 0.013 | 0.665 +/- 0.013 |
| Test accuracy | 0.651 +/- 0.0017 | 0.650 +/- 0.002 | 0.655 +/- 0.0016 | 0.656 +/- 0.0015 |

According to these results and using the validation accuracy as the criterion, the logistic model using the elastic-net showed better accuracy, although not statistically significant. Since there was still bias error, a more complex model was needed to show improvements.

More complex models are: NN, SVM, XGBoost, Decision Trees

For each model

**SVM**

SVM is a more complex model compared to logistic regression – it optimizes the distance between data points and the decision boundary. The hyperparameters of three different kernels were optimized. The linear kernel was the simplest kernel and it showed a training accuracy of 0.598 +/- 0.0019, a validation accuracy of 0.592 +/- 0.034, and a test accuracy of 0.588 +/- 0.0041. This model had very little variance error and significantly worse bias error than the logistic regression model.

To improve on this and reduce the bias error, the more complex Gaussian (rbf) and sigmoid kernels were used.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Linear kernel | RBF kernel | Sigmoid kernel |
| Training accuracy | 0.598 +/- 0.0019 | 0.656 +/- 0.0017 | 0.663 +/- 0.0026 |
| Validation accuracy | 0.592 +/- 0.034 | 0.660 +/- 0.012 | 0.658 +/- 0.022 |
| Test accuracy | of 0.588 +/- 0.0041 | 0.649 +/- 0.0013 | 0.652 +/- 0.0026 |

Using these SVM models did not show any improvements compared to the logistic regression model.

**Artificial Neural Networks**

These ANNs only contained a single hidden layer, as initial hyperparameter testing consistently chose smaller networks. As can be seen from the tables, the highest prediction accuracy was in the 15 game dataset, on the regularized networks. This scenario yielded a full percentage point more accuracy than the unregularized networks which indicates that the regularization did improve performance. This accuracy of ~65% is in line with the results that we have seen in previous literature. Next steps to improve the accuracy will be outlined in the Future Work section.

|  |  |  |
| --- | --- | --- |
| 5 games | Unregularized | Regularized |
| Training Accuracy | 0.641 +- 0.003 | 0.637 +- 0.014 |
| Validation Accuracy | 0.646 +- 0.027 | 0.631 +- 0.035 |
| Testing Accuracy | 0.619 +- 0.003 | 0.614 +- 0.012 |

|  |  |  |
| --- | --- | --- |
| 10 games | Unregularized | Regularized |
| Training Accuracy | 0.651 +- 0.002 | 0.653 +- 0.002 |
| Validation Accuracy | 0.647 +- 0.010 | 0.651 +- 0.017 |
| Testing Accuracy | 0.641 +- 0.002 | 0.639 +- 0.002 |

|  |  |  |
| --- | --- | --- |
| 15 games | Unregularized | Regularized |
| Training Accuracy | 0.652 +- 0.002 | 0.661 +- 0.002 |
| Validation Accuracy | 0.650 +- 0.014 | 0.661 +- 0.011 |
| Testing Accuracy | 0.642 +- 0.002 | 0.652 +- 0.003 |

XGBoost

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| XGBoost Results | | Unnormalized | | Normalized | |
| Unregularized | Regularized | Unregularized | Regularized |
| 5-game | Training Accuracy | 0.707 ± 0.005 | 0.704 ± 0.006 | 0.696 ± 0.003 | 0.678± 0.003 |
| Validation Accuracy | 0.632 ± 0.012 | 0.611 ± 0.017 | 0.632 ± 0.013 | 0.638 ± 0.021 |
| Test Accuracy | 0.614 ± 0.005 | 0.602 ± 0.007 | 0.620 ± 0.003 | 0.620 ± 0.004 |
| 10-game | Training Accuracy | 0.747 ± 0.004 | 0.748 ± 0.004 | 0.670 ± 0.006 | 0.666 ± 0.005 |
| Validation Accuracy | 0.636 ± 0.13 | 0.646 ± 0.014 | 0.640 ± 0.022 | 0.628 ± 0.016 |
| Test Accuracy | 0.629 ± 0.004 | 0.632 ± 0.005 | 0.629 ± 0.007 | 0.627 ± 0.008 |
| 15-game | Training Accuracy | 0.682 ± 0.003 | 0.696 ± 0.004 | 0.761 ± 0.008 | 0.729 ± 0.003 |
| Validation Accuracy | 0.644 ± 0.012 | 0.666 ± 0.012 | 0.659 ± 0.017 | 0.648 ± 0.022 |
| Test Accuracy | 0.639 ± 0.004 | 0.647 ± 0.003 | 0.646 ± 0.003 | 0.650 ± 0.003 |

-discuss hyperparameters

-discuss first results and bias-variance

-discuss model adjustment (ex. Addition of regularization or more complex kernels)

-discuss final bias-variance

Final bias-variance stuff

* Very little variance error (SVM and log reg)
* Significant bias error (accuracies at about 65-66%)
* Need more features to reduce bias error - feature mapping etc.
* Regularization reduces variance error, so we don’t have a real need for it

Final Test

-comparison of all test accuracies and precision-recall comparison -

-what’s the best model?

Next Steps

-ensemble model

-more features = more money -> being correct 65-70% of the time is kinda dope, as long as bets are placed correctly a lot of money could be made (I’m still down to try this) LOL

Gotta use that OSAP money for it though lol

**Implementation and Code (0.5 pgs)**

**EVERYTHING BELOW THIS POINT IS READY FOR LATEX**

**FINAL REPORT**

**Introduction (0.5-1pg) (subheadings for organization, will be removed after)**

-Problem (Honestly didn’t know what to say so I made it cheesy, please review/edit)

Sports are a favourite pastime for people around the world. This global popularity has resulted in a huge number of statistics being collected and organized, a goldmine for data scientists. This makes predicting the outcome of sporting events both a fun pastime, and a lucrative one when it comes to sports betting. Using Machine Learning (ML) for predictive sports modeling is a very well documented field, with basketball being the second most studied sport [1].

-Significance and Challenge

It has been shown that ML can be used to generate a profit from sports betting [5]. In this case the greater the accuracy of the models the greater the potential profit. However, even with the amount of current documentation that exists, there appears to be a glass ceiling of 75% accuracy in current models [6]. This could stem from the fact that, even with the varying amount of models that have been used, similar features are being selected. Feature selection has been argued to be even more important than model selection in this problem. Another issue is that data sets vary drastically from study to study, inhibiting researchers from constructively comparing their findings with previous work and leading to unclear development [7].

-Prior Work (didn’t have much on past work in the proposal)

Datasets for basketball statistics are maintained by the NBA on their website, as well as multiple Kaggle datasets which fit the needs of this project. The most popular models for sports prediction tend to be artificial neural networks and support-vector machines, with the latter performing the best in many instances [3][4].

-Overview of Report

A dataset from something or another was selected for this project, due to the specific statistics it had. The statistics were averaged for each team over the last 5, 10, and 15 games. These averages were separated into three different datasets for comparison. ANOVA tests were conducted on the datasets to select the top 15 features. Once those features were extracted, the final dataset was normalized. Initially, logistic regression was used as a baseline model. More complex models were selected from the literature to be: Neural Networks, SVM, Decision Trees, and XGBoost. All models were tuned and iterated before a final test was run for comparison.

**Problem Statement and Dataset (0.5 -1pgs)**

**Problem Statement**

There is plenty of compiled data made public by professional sports organizations such as the NBA. This data can be used to predict the outcomes of future matchups which can benefit parties such as sports analysts, ticket sales, fans, and those involved in sports betting. The goal of this study is to analyze the performance of different machine learning algorithms and select our optimum algorithm for predicting match outcomes.

**Dataset**

With the vast collection of extensive data in professional basketball found on websites such as the Official NBA website, Basketball Reference, and ESPN, we needed to compile data that was appropriate for the modern game. This data collection dated back to the middle of the 20th century with more detailed player level statistics, advanced performance metrics, etc. starting to appear in the 1980s. To predict team performance, we needed a dataset with these more detailed statistics over many seasons.

We narrowed our research down to the NBA Enhanced Box Scores and Standings (2012-2018) database from Kaggle ([https://www.kaggle.com/pablote/nba-enhanced-statistics](https://www.kaggle.com/pablote/nba-enhanced-stats)), which includes data for every regular-season game starting from the 2012-2013 season, up until the 2017-2018 season. This dataset was created on Java using RESTful APIs provided by xmlstatistics (https://erikberg.com/api). It includes player-level statistics for each game, down to the quarter. It also includes detailed team-level statistics, which tally the player-level data for each game, and calculaute advanced statistics such as Efficiency Differential, and Offensive/Defensive ratings which proved to be valuable (will be discussed in more detail in the feature selection section). Moreover, the 2012-2018 period is representative of the dynamics of today’s NBA as it includes the majority of today’s most dominant teams and players. We also had over 7500 games to train our models on, which was sufficient and produced results that are in-line with our literature review.

The team box-score data is laid out where there are two rows per game, one row for the home team, and one row for the away team. Each row had two sets of columns with team statistics and opponent statistics (as in the statistics for each game was repeated twice from different perspectives, which is useful for some sports-modeling).

**Methods and Models (1-2pgs)**

**Data Engineering**

We needed to get the data in a format where every row presents the rolling average of statistics for either team in the match-up over the past n games. This smooths the data and removes noise for implementing our classification models. First, the data was sorted based on game date and the team IDs. A dictionary was created that was indexed by team ID and game date to store the raw data from the dataset. An additional key was added for the win percentage (at this point, before averaging this consisted of ones and zeroes for wins and losses, respectively). Irrelevant columns, such as team division, and referee names were disregarded in our model.

To acquire a statistics average for the last n games, this ordered dictionary was required. The statistics were averaged for the last 5, 10, and 15 games in separate datasets. For each team ID, an average was taken at each game date consisting of the previous n games; this data replaced the dictionary row containing the raw data – the only parts of the data untouched were the win or loss column (to be used for the validation labels, which our algorithm generated). These averaged data points then replaced the original non-averaged row in the initial data frame where the data was first read. This resulting dataset was then in a form that we didn’t want to work with. In this form, the data read such that each game had two rows: one for the home team with the statistics of the previous opponents that they had faced, and one for the away team of a game with the previous statistics of the opponents they had played.

We tackled this by going into the original dataset and deleting the opponent columns entirely. This left us with only one column of data, that had a row of home team data and away team data for each game. Running our original averaging algorithm on this version of the dataset meant that for each row, we now have an average of how that specific team performed over the last n games. We then sorted the dataset manually by “Home”/”Away”, followed by sorting each category by game date, followed by concatenating both blocks in a way that left us with one row per game – a set of columns with home statistics, and a set of columns with away statistics (this is why we had to recalculate pace and possession). We also came up with an alternative algorithm, where the original program was amended to average home and away performances for each team exclusively, but this version of the dataset proved to be redundant.

**Feature Selection**

This task was dependent on the features that are present in the dataset. In this case, we have a set of numerical input features with a binary categorical output. Because of this combination we used ANOVA testing to determine the correlation between each feature and the output. The feature selection algorithms from the package scikit-learn were used to perform this task. Once the correlation between features was visualized, the top 15 features were selected to be used as the inputs to our models. The number 15 was used after a visual inspection of the correlations, as 15 seemed to encompass all of the highly correlated features as well as a few less correlated features.

**Data Normalization**

Algorithms that rely on combining/manipulating features through feature mapping can run into issues when there is a large difference in the scale between features. This also presents a problem when updating weights through gradient descent as unscaled data can cause large errors for initial parameter guesses in the update rule resulting in an unstable learning process.

<https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/>

<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/normalize-data>

**Model Selection**

Models were selected based on what was found in relevant literature. ANNs and SVM have been extensively used in this domain previously \cite{3} \cite{4}. Decision trees have been applied to other applications in the sports domain, namely predicting the hall of fame status of Major League Baseball players. We applied decision trees to our problem because we wanted to see if it was efficient in another sports domain. Similarly to decision trees, gradient boosting was chosen because of its efficiency in predicting English Premier League Team Season Win Percentages \cite{}. We thought that this application was similar to our problem and wanted to include it for that reason.

**Model Implementation**

For this project, models were implemented from pre-existing libraries such as scikit-learn and Keras. For a baseline model, logistic regression was chosen due to the problem being binary classification. The more complex models that were implemented are: Decision Trees, XGBoost, SVM, and Neural Networks. For each model the dataset was split into a ratio of 70/10/20 for training/validation/test datasets. Optuna was used to tune the hyperparameters of each model, with the objective of maximizing the validation accuracy. This was done for each dataset (rolling 5, 10, and 15 game averages). Once a tuned set of hyperparameters was obtained for each situation, the models were trained on the training dataset and ran predictions on the validation set. This training/predicting process was run 10 times for each situation to provide statistical weight behind the results. The average training and validation accuracies were compared for bias-variance analysis to determine if regulation needed to be added/reduced or if a more complex model was needed. The model was then re-tuned with the added changes and the bias-variance analysis was done on the new results. Once a model was finalized it was trained on the training dataset and ran predictions on the test data set for the final results. This was also done 10 times and average for statistical significance. For the final comparison between models, precision and recall were compared in addition to accuracy.

**Results and Discussions (1-2pgs)**

**Log Reg**

The first model tested for this problem was a logistic regression model without regularization using up to 10000 epochs and a tolerance of 0.0001. The 15-game dataset using n=10 independent trails was the most accurate with an accuracy of 0.661 +/- 0.0021 on the training set, an accuracy of 0.665 +/- 0.014 on the validation set, and an accuracy of 0.651 +/- 0.0017 on the test set. As this was the simplest model, this served as a baseline for the more complex models. Based on the similarity between the validation set and the training set, there was very little variance error. However, at 65 percent accuracy, there was significant bias error.

To find improvements in this model, three types of regularization were applied – an L1 regularizer, an L2 regularizer, and an elastic-net regularizer (a combination of L1 and L2). The hyperparameters for these models were tuned, and the following accuracies were obtained for the 15-game dataset. According to these results and using the validation accuracy as the criterion, the logistic model using the elastic-net showed better accuracy, although not statistically significant. Since there was still bias error, a more complex model was needed to show improvements.

**SVM**

SVM is a more complex model compared to logistic regression – it optimizes the distance between data points and the decision boundary. The hyperparameters of three different kernels were optimized. The linear kernel was the simplest kernel and it showed a training accuracy of 0.598 +/- 0.0019, a validation accuracy of 0.592 +/- 0.034, and a test accuracy of 0.588 +/- 0.0041. This model had very little variance error and significantly worse bias error than the logistic regression model.

To improve on this and reduce the bias error, the more complex Gaussian (rbf) and sigmoid kernels were used. Using these SVM models did not show any improvements compared to the logistic regression model.

**XGBoost**

XGBoost (Extreme Gradient Boosting) is a popular algorithm that uses gradient boosted decision trees. There are quite a few hyperparameters that can be fiddled with in the XGBoost library. For this application we set the objective to be classification with the softmax objective, and 2 classes (win or lose). The other hyperparameters of interest (max depth, learning rate, gamma, minimum child weight and epochs) were tuned using Optuna to maximize the validation accuracy. The tuned model was then run 10 times to train and predict for computation of training and validation accuracies. Decision Trees algorithms do not require normalized data, so this was done with all 6 datasets, with the 15-game normalized dataset having the highest validation accuracy of 0.6589942 ± 0.01684119. Compared to a higher average training accuracy of 0.76090792 ± 0.00818973, this is indicative of high variance possibly due to overfitting. To counter this, the model was returned with the inclusion of the L1 and L2 regularization weights, alpha and lambda. This resulted in a model with average training accuracy of 0.72934596 ± 0.0034167 and average validation accuracy of 0.64758221 ± 0.02238706 which shows slightly less variance than the un-regularized model. (something about random forest maybe being better since it uses bagging instead of boosting, will look at Ali’s results first or mention in comparison).

**Artificial Neural Networks**ANNs have several important hyperparameters that must be tuned before an ANN can achieve acceptable performance. In the ANN these hyperparameters include: number of hidden layers, size of hidden layer(s), activation function(s), learning rate, and the optimization algorithm. In our work we used Optuna to tune these parameters. We also included some regularization in our networks as a comparison to the original networks. Specifically L1 and L2 regularization was applied. The un-regularized models were found to be acceptable - they did not have extremely high accuracy, with the highest being ~64% for both 10 and 15 game averages. The regularization of these models were found to increase performance slightly, just as planned. In this case the accuracy only increased for the 15 game average, by about 1% to ~65%. This shows that even though we were not able to get an extremely accurate model, the regularization was still able to improve the performance.

**Discussion**

With the models finalized, they were run on the test dataset 10 times to get the average accuracies. The models in order from highest to lowest test accuracy were: (list). While XX was the highest model accuracy, it was not much higher than the baseline logistic regression model. In fact, all model accuracies were close together. Looking at the training and validation accuracies we can see that they are similar for each model, meaning that the models are suffering from bias. Unfortunately using the more complex models did not prove to decrease the bias that the logistic regression model was suffering from. This confirms what was speculated earlier, that the feature selection is what is limiting the prediction accuracies for this problem. We were able to achieve accuracies close to what was seen in the literature so this remains a consistent problem.

**Implementation and Code (0.5 pgs)**

The code used to do each of the steps in this work, data engineering, feature selection, normalization, hyperparameter tuning, and final tests was implemented in Python. Data engineering was completed using a combination of Pandas and Numpy, where data was loaded using Pandas, and a Numpy list was created that stored the merged data. Feature selection and data normalization was completed using scikit-learn, namely the SelectKBest, Normalizer, and f\_classif functions. These steps were performed once the data engineering steps were completed. Finally the data was saved using Pandas. Algorithm implementations were done using scikit-learn, the XGBoost library, and Keras for the ANNs. This allowed us to focus more on the problem at hand rather than ensuring that our implementation was correct. Hyperparameter tuning was performed using the Optuna package. This package allowed us to create numerical or categorical options for the package to select from using Python syntax. Therefore individual parameters of each algorithm can have their own range of values to select from. Once this was completed for each algorithm, non-regularized and regularized, the final tests were completed. Once again these tests were completed using scikit-learn or Keras, depending on the algorithm.