STAT 441 Final Report

Mark Chen, Reven Liu, Arvind Nagabhirava, Rain Zhao

2023-04-24

Contents

1 Summary

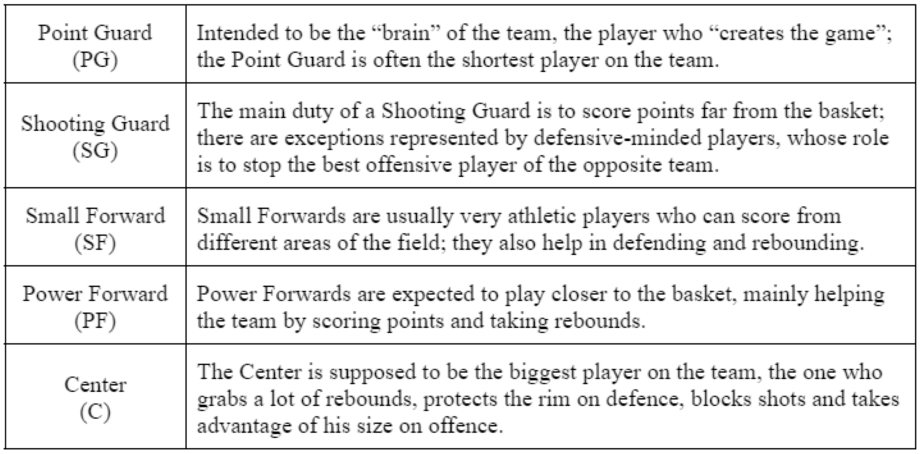
This project used multiple classification methods to investigate whether an NBA player's season statistics can be used to ascertain the position they play. The aim was to predict a player's position in the 2022-2023 season based on training data from the prior 21 seasons. We applied support vector machines (SVMs), tree-based bagging, artificial neural networks, and developed a hybrid SVM-tree model to overcome the limitations of typical decision trees. A filter method was applied to identify relevant predictors across the models and k-fold cross validation was used across all models for hyperparameter tuning. After training on 2001-2021 data, models were tested on the test set of the 2022-2023 NBA season. Test results showed that the support vector machine with a linear kernel performed the worst, while the hybrid model and artificial neural network approaches were the best performing models—successfully classifying 83.1% of player positions for the 2022-2023 NBA season. While the training and hyperparameter tuning process for the latter methods were more computationally intensive, they resulted in better performance on both the training and test sets.

2 Introduction

The game of basketball traces its roots to Springfield, Massachussets in 1891 and was invented by James Naismith. The game has changed quite a bit over its history but one of the things that has remained constant across at least the past 75 years are the 5 key positions of the game in the figure below.

The motivation is to investigate whether a player's season statistics can be used to predict their position. The timing for this project was perfect since the current regular season ended on April 9th, 2023 and the development of the classification methods listed below began before the season had ended!

Decision Trees with Bootstrap Aggregation, Support Vector Machines (SVM), an SVM/Tree Hybrid model, and an Artificial Neural Network were all implemented to carry out this classification task.



Point Guard (PG)	Intended to be the "brain" of the team, the player who "creates the game"; the Point Guard is often the shortest player on the team.
Shooting Guard (SG)	The main duty of a Shooting Guard is to score points far from the basket; there are exceptions represented by defensive-minded players, whose role is to stop the best offensive player of the opposite team.
Small Forward (SF)	Small Forwards are usually very athletic players who can score from different areas of the field; they also help in defending and rebounding.
Power Forward (PF)	Power Forwards are expected to play closer to the basket, mainly helping the team by scoring points and taking rebounds.
Center (C)	The Center is supposed to be the biggest player on the team, the one who grabs a lot of rebounds, protects the rim on defence, blocks shots and takes advantage of his size on offence.

Figure 1: 5 key basketball positions.

Data Retrieval

The data was taken from the ESPN NBA Player Stats website (https://www.espn.com/nba/stats/player/_/season/2022/seasontype/2). An important note regarding this data is that players are only listed in this dataset if they played at least 70% of their team's games in a given season. Since a majority of the statistics listed per player are aggregated statistics across a season, it is important that the sample size per player is large enough for this average value to be a robust measurement. For example, the Points Per Game statistic is an increasingly robust measure of a player's scoring ability the larger the number of games the player has played in that season.

Combining Positions

It is important to note that players will be classified into two groups of positions: guards (point guards, shooting guards, and guards, n=3858) and forwards (small forwards, power forwards, forwards, and centers n=5551). There are significantly more forwards than there are guards since three out of the five positions are forwards. This grouping reduces the number of classes from 7 to 2 and reflects the groupings used in other studies (Yolanda Escalante 2010 ; Jaime Sampaio 2006). Further motivation for this split is to increase the number of training examples available for each class. Finally, methods such as SVM and our idea for an SVM-Tree Hybrid are best suited to a binary classification problem.

Notes on the Data

No Position Listed:

The only player in the retrieved data with no listed position is Eddy Curry, who played in 10 seasons between 2001 and 2013. Our own research showed that Curry played as a Center and the data was updated to reflect this.

Guard/Forward

There's only one player that is listed as a GF—both a Guard and a Forward, Jiri Welsch. For the purposes of this classification, we will ignore Welsch since we are interested in classifying players as either a Guard or a Forward, not both.

The Lockout Season

The 2011-2012 NBA season saw the fourth lockout in the history of the NBA following disputes between the NBA and the National Basketball Players Association (Thomas Sadler 2016). This both delayed the season and reduced its length from 82 games to 66 games. Although the statistics retrieved from the ESPN website are per-game statistics and not total statistics, the 2011 season data is being discarded due to its shorter season length. Since players are known to hit their stride as both individuals and as a team as the season progresses, it is possible for a shorter season to lead to systematically misrepresented statistics across all players as compared to a regular season and thus the 2011 season data will not be considered for this classification task.

Exploratory Analysis

From the following plots, we notice that forwards are more likely to have higher statistics for rebounds and blocks whereas guards are more likely to have higher statistics for assists and shooting statistics such as field goals attempted, free throw percentage, and three points attempted.

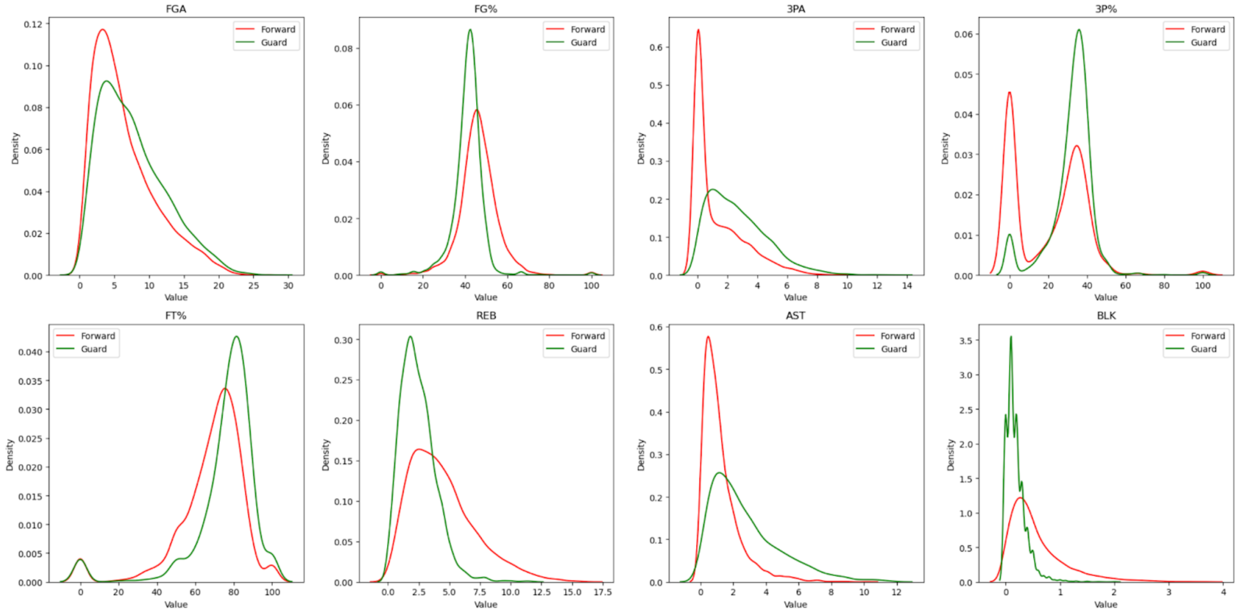


Figure 2: Feature distributions amongst guards and forwards. A complete plot of all features as well as a definition of each feature is available in the Appendix section 5

The radial plots express features as a percentage of the maximum observed amount for each feature. These plots further emphasize the difference between guards and forwards, with forwards leading significantly in rebounds and blocks and guards in assists.

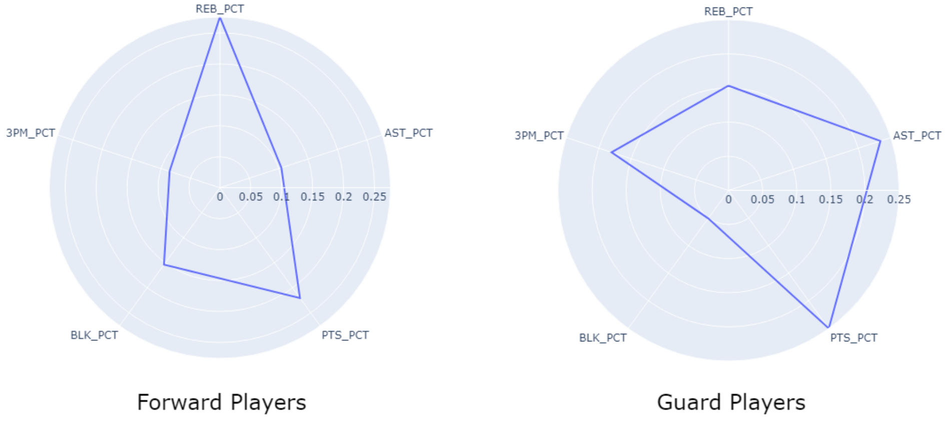


Figure 3: Radial Plot of Features.

3 Methods

Preprocessing

Two-Pointer (2P) Statistics

The scoring statistics shown in the ESPN dataset are broken down into field goals, three-pointers, and free throws. However, it is important to note that since field goals are comprised of all non-free throw points scored by a player, three-pointers are a subset of field goals. Under the hypothesis that larger players are often more adept at scoring from within the three-point line in the form of layups, dunks, and midrange jump shots and larger players are more likely to be forwards than guards, we decided to derive the following statistics to help improve our classification efforts:

- 2PA: The number of two-pointer scoring attempts per game
- 2PM: The number of two-pointers scored per game
- 2P%: The scoring percentage of two-pointers per game

Using the 2P statistics instead of the field goal statistics also provides the advantage of reducing the multicollinearity amongst the features since the three-pointer statistics are a subset of the field goal statistics.

Feature Selection

For feature selection, a filter method was used. In the context of feature selection, filter methods select features irrespective of the models being used. Instead, attributes such as the correlation between the predictor and response are considered. As a result, filter methods provide features that are generalizable to many models, but not specifically chosen for any particular model. We computed the mutual information with the SelectKBest function, which finds the k best predictors based on the mutual information scores. Mutual information measures the amount of shared information between two variables, also known as the Kullback-Leibler divergence. The formula:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} P_(x,y) log(\frac{P_(x,y)}{P(x)P(y)}) \qquad (1)$$

Below is the plot of mutual information scores for all variables with a score greater than zero:

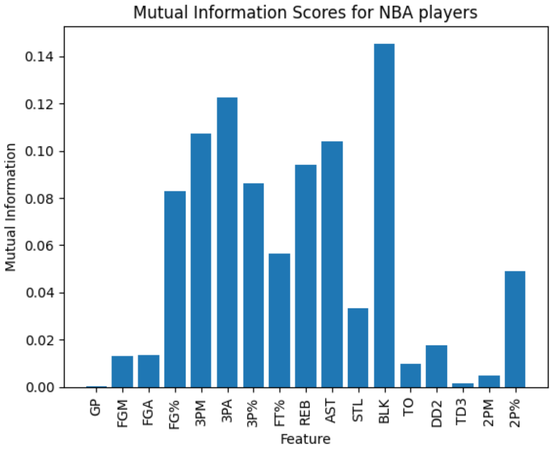


Figure 4: Plot of Mutual Information

Using the top k = 10 best features based on mutual information, all of our models (except the artificial neural network) are trained on the features FG%, 3PM, 3PA, 3P%, FT%, REB, AST, STL, and BLK.

Classification Approaches

1. Support Vector Machine (SVM)

Linear, radial basis function (RBF), and polynomial kernels were considered for SVM, resulting in three separate SVM models. Each model was tuned using 5-fold cross-validation with grid search to find the optimal value of C, the tradeoff between margin size and misclassification, over the values [0.01, 0.1, 1, 10]. Similarly, the RBF and polynomial kernel SVMs were tuned for the optimal setting of gamma over [‘scale’, ‘auto’].

These values were found to be C = 0.1 for the linear kernel, C = 10 for the RBF and polynomial kernels, and gamma = ‘scale’ for the RBF and polynomial kernels.

2. Decision Tree - Bootstrap Aggegration

A bagging classifier was used with a decision tree as the base estimator. For the decision tree, 5-fold cross-validation was used to tune the hyperparameters using randomized search to find their optimal values. Randomized search functions similarly to grid search, but with samples of combinations instead of an exhaustive search. The results of the randomized search were as follows:

Loss criterion over [‘gini’, ‘entropy’, ‘log_loss’] was found to be log loss.
Minimum samples to split over [2,4,6,8] was found to be 8.
Minimum samples for a leaf node over [1,2,3] was found to be 1.
Maximum features for a split over [sqrt, log2] was found to be log2.
Maximum depth over [10, 20, 30, 40, 50] was found to be 10.

The pseudocode for the randomized search algorithm is as follows:

```
for i in n_iterations:
   1. Select a random sample of hyperparameters
   2. Fit the model using the selected hyperparameters and evaluate the
      performance using .score() method
   3. Store the result
Return the best-performing model.
```

3. SVM-Tree Hybrid Model

This hybrid approach aims to overcome one of the key limitations of traditional decision trees, the tendency to fit overly complex boundaries on linearly and near-linearly separable data.

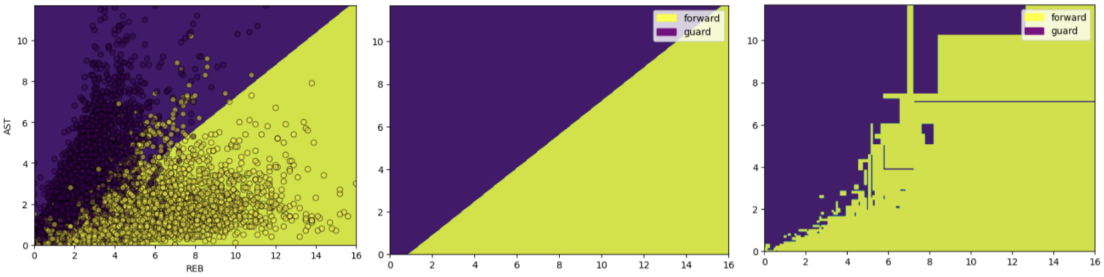
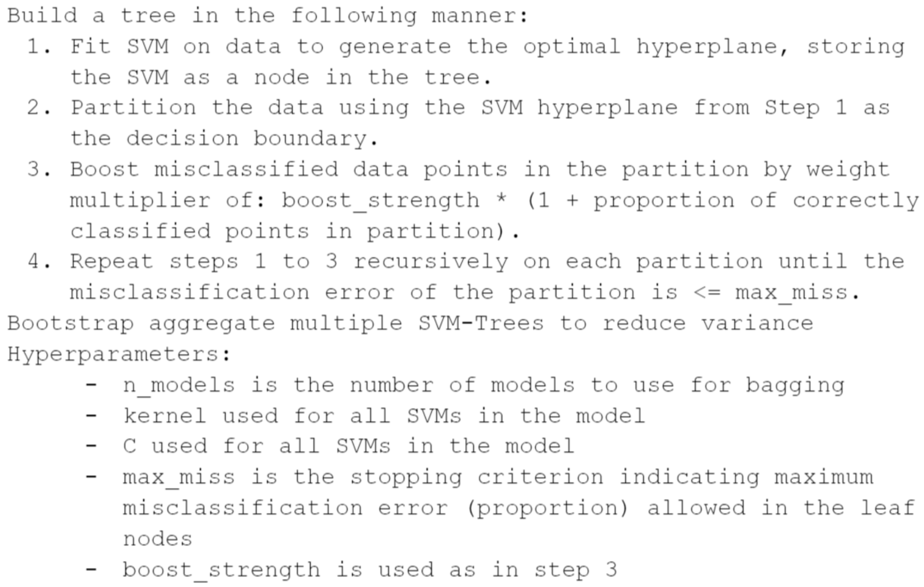


Figure 5: Left: NBA dataset projected down to two features: REB (number of rebounds per game) and AST (number of assists per game). The data scatter is overlaid on the SVM-Tree decision boundary. Center: Decision boundary of a 2 layer SVM-Tree trained on the two features. Right: Decision boundary of a 10 layer decision tree trained on the two features.

A hybrid approach that allows each split in the decision tree to be a hyperplane generated via SVM has been researched and the results are promising (Fumitake Takahashi 2002). Our approach in particular executes the following algorithmic sketch:



```
Build a tree in the following manner:
  1. Fit SVM on data to generate the optimal hyperplane, storing
     the SVM as a node in the tree.
  2. Partition the data using the SVM hyperplane from Step 1 as
     the decision boundary.
  3. Boost misclassified data points in the partition by weight
     multiplier of: boost_strength * (1 + proportion of correctly
     classified points in partition).
  4. Repeat steps 1 to 3 recursively on each partition until the
     misclassification error of the partition is <= max_miss.
Bootstrap aggregate multiple SVM-Trees to reduce variance
Hyperparameters:
        -  n_models is the number of models to use for bagging
        -  kernel used for all SVMs in the model
        -  C used for all SVMs in the model
        -  max_miss is the stopping criterion indicating maximum
           misclassification error (proportion) allowed in the leaf
           nodes
        -  boost_strength is used as in step 3
```

The single SVM-Tree model was tuned using 5-fold cross validation grid search on the following hyperparameters:
Kernel=linear, C in [0.1, 1, 10, 100],
Max_miss in [0.001, 0.01, 0.1],
Boost_strength in [1, 10, 100, 1000].

The optimal hyperparameters were C=1.0, max_miss=0.001, boost_strength=100 and used to evaluate the single SVM-Tree model in the results section below. Similarly, the SVM-Tree Bagging model was tuned using 5-fold cross validation grid search on the following:
n_models=10, kernel=linear,
C in [0.1, 1, 10, 100, 1000],
max_miss in [0.001, 0.01, 0.1, 0.2],
boost_strength in [1, 10, 100].
This took 864 seconds and the optimal hyperparameters were C=1000, max_miss=0.1, boost_strength=1. Furthermore, these optimal hyperparameters were fixed to tune n_models separately using 5-fold cross validation on n_models in [10, 20, 40, 80, 160] with the optimal n_models being found to be 160, as shown in Figure 6.

4. Artificial Neural Network (ANN)

5 fold cross-validation was used to determine the number of hidden layers and the number of neurons per layer. For this dataset, cross-validation showed that one hidden layer containing 24 neurons produced results equal in performance to models with two or three layers and performed just as well as models with 48 and 96 neurons in the hidden layer.

Feature selection for the ANN is notably different than the other classifiers concerned here since unimportant features will be automatically down-weighted by the weight updates during the back-propagation algorithm. As a result, all features in the original dataset were used to train the neural network with the exception of the player's name.

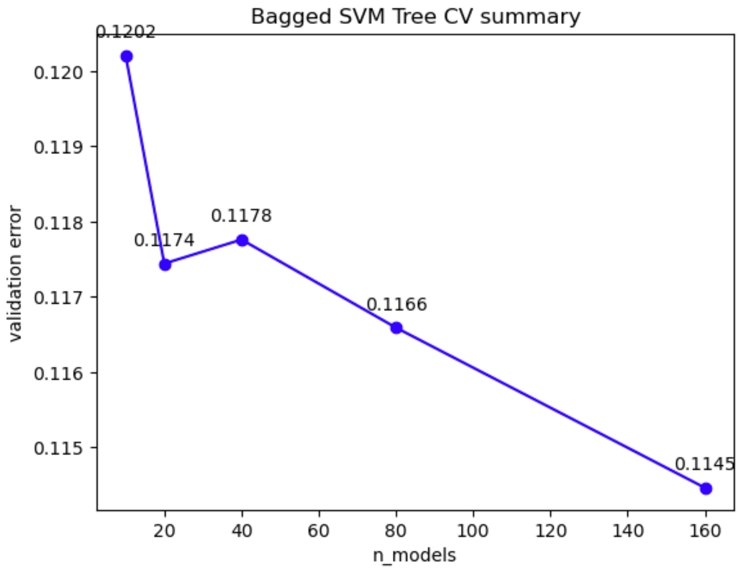


Figure 6: SVM Hybrid Number of Models vs. Validation Error

4 Results

The test set contains 534 rows of NBA player statistics for the 2022-2023 season. Results are summarized in the table below:

Model	Parameters	Error	Calibration Timing	Model Fit Runtime
SVM - Linear Kernel	0.130	0.200	5s	0.06s
SVM - RBF Kernel	0.112	0.178	79s	4s
SVM - Polynomial Kernel	0.125	0.184	96s	6s
Tree-based Bagging	0.076	**0.169**	8s	1s
Single SVM-Tree	0.125	0.182	97s	0.12s
SVM-Tree Bagging	0.050	**0.169**	1594s	88s
Artificial Neural Network	0.107	**0.169**	280s	45s

With the training set of approximately 9500 rows, the neural network, SVM-Tree bagging and tree-based bagging models performed equally the best at a 16.9% misclassification error on the test set. From the generated confusion matrices below, the misclassified observations are split up roughly evenly between forwards and guards with guard being misclassified more than forward for all three of the aforementioned models. The full set of confusion matrices is available in Appendix section 5.

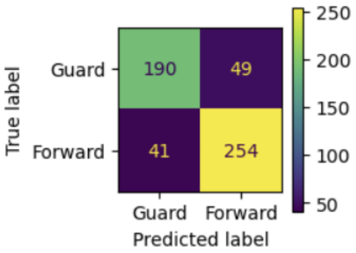


Figure 7: Confusion Matrix for Hybrid-SVM Model

Comparing Classifiers

The linear SVM performed the worst on the test set, misclassifying 20% of the players. SVM with RBF and polynomial kernels outperformed the SVM with a linear kernel—likely due to the increased flexibility stemming from their non-linear decision boundaries. The single SVM-Tree was able to outperform linear SVM on the test error by 1.8 percentage points. Finally, the fitted SVM-Tree had just 3 layers, and as the projection plots indicate below, the most significant split at the root and was nearly identical to the linear SVM split, as expected.

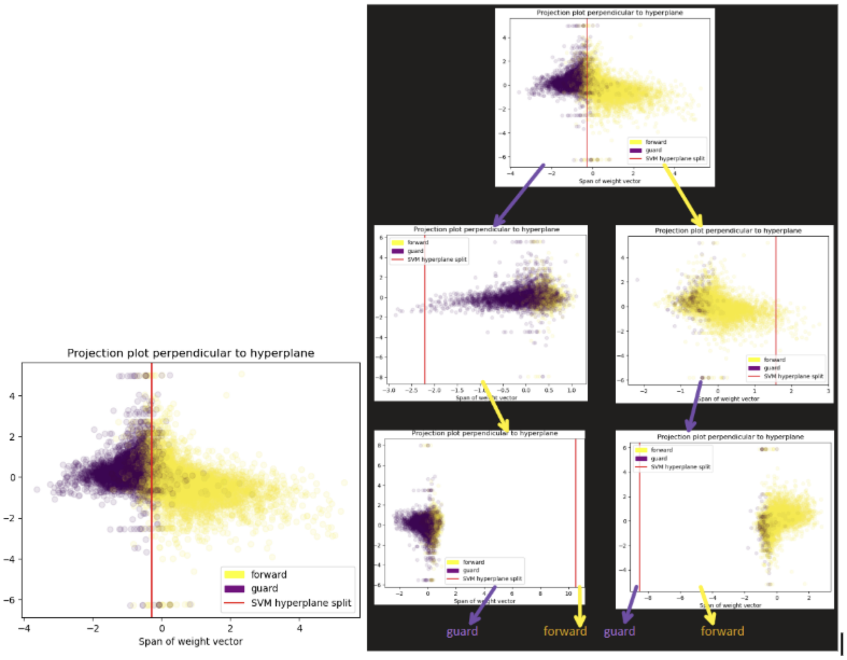


Figure 8: Left: projection plot of linear SVM indicating hyperplane split by red vertical line. Right: projection plots of the nodes of the SVM-Tree organized into the shape of the tree with yellow and purple arrows indicating the forward and guard splits respectively for each node.

The 80% accuracy of linear SVM and the above plots indicate that the data is near-linearly separable. Furthermore, the deeper splits in the SVM-Tree are unable to make much progress in this dataset beyond the initial split, hence we don't see a drastic improvement over the linear SVM model.

One of the goals for the SVM-Tree model was to reduce the complexity of the decision tree. Trees in the bagged decision tree approach had 508 splits on average whereas the bagged SVM-Tree model had 131 and 119 splits on average when bagging with 160 trees and 10 trees respectively. Thus, we have succeeded in reducing the complexity of the decision trees by allowing hyperplane splits via SVM. Remarkably, the bagged SVM-Tree approach achieved the same test error as an Artificial Neural Network which can be credited to both the ensemble learning and the aforementioned advantages of applying SVM splits to decision trees.

5 Discussions

Limitations of our SVM-Tree Hybrid Approach

Many researchers have studied hybrid models using SVM splits in decision trees. In a study conducted at the Rensselaer Polytechnic Institute, researchers explored how different optimization algorithms would affect a decision tree model which uses SVM splits to formulate the nodes. Their findings indicate that the "GTO/SVM [Global Tree Optimization/Support Vector Machines algorithm] works better than GTO alone and produces much simpler trees than conventional DT algorithms" (K. P. Bennett 1998). The GTO/SVM algorithm optimizes a fixed size SVM-Tree, tuning all the SVMs together under a single objective function as seen in Figure 9.

In contrast to (K. P. Bennett 1998), our algorithm optimizes each SVM split individually and myopically,

$$
\begin{array}{lll}
\displaystyle\min_{w,\eta,\nu,\tau} & \frac{\lambda}{2}[(w^0 \cdot w^0) + (w^1 \cdot w^1) + (w^2 \cdot w^2)] & \\
 & \displaystyle (1-\lambda)\left[\sum_{i=1}^{m}\eta_i + \sum_{j \in D^1}\nu_j + \sum_{k \in D^2}\tau_k\right] & \\
s.t. & q_i[x_i \cdot w^0 - b^0] \geq 1 - \eta_i & \eta_i \geq 0 \\
 & & i = 1, \ldots, m \\
 & t_j[x_j \cdot w^1 - b^1] \geq 1 - \nu_j & \nu_j \geq 0 \\
 & & j \in D^1 \\
 & t_k[x_k \cdot w^2 - b^2] \geq 1 - \tau_k & \tau_k \geq 0 \\
 & & k \in D^2
\end{array}
$$

Figure 9: From the Bennett 1998 paper, we see the objective function for a 2-layer tree with 3 SVM nodes.

which may cause the entire SVM-Tree to be suboptimal. Although we did not implement the optimization algorithms mentioned in the above study, our findings are consistent with the Rensselaer Polytechnic Institute study in that the hybrid SVM-Tree approach would generate simpler boundaries and thus produce more generalized and robust models.

Improvements of our Hybrid Approach to SVM-Trees

A notable area of improvement would be to reformulate the optimization problem as a Global Tree Optimization as done by (Bennett 1995) and (K. P. Bennett 1998). The benefit being that each SVM split would be optimizing for the fit of the whole tree rather than optimizing for its local partition of the data, which may be suboptimal in the long run. (K. P. Bennett 1998) also formulated a dual problem of the SVM-Tree with support for a kernel function for each SVM split.

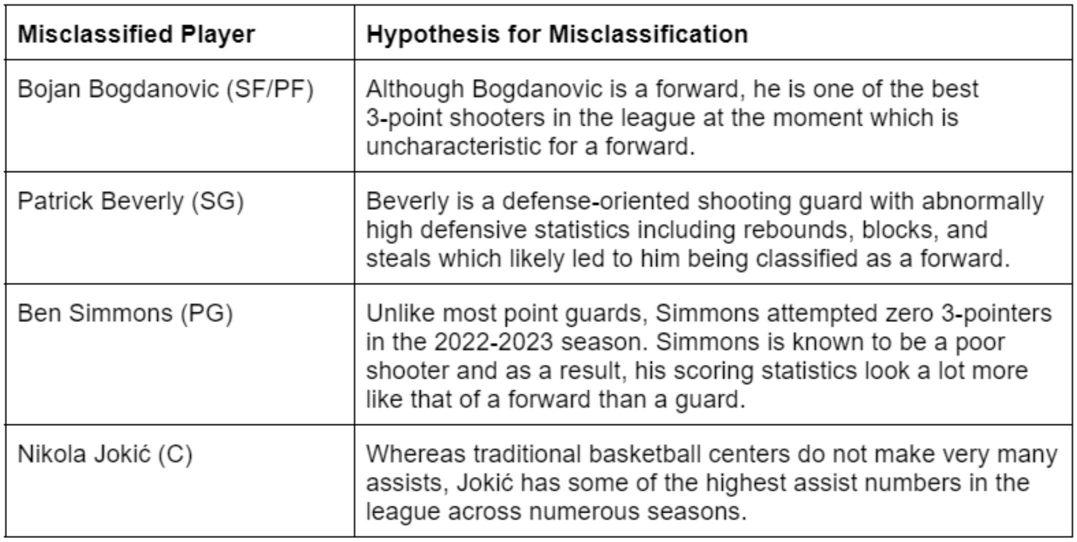
Another area for exploration is the use of non-linear kernels for the SVM splits within our hybrid approach. Although the use of such kernels would be more computationally expensive, we believe they could lead to even more robust models. Finally, there is potential for a more thorough tuning of the hyperparameters of our SVM-Tree.

Systematic Changes

Due to rule changes and new playing styles in the NBA, basketball has changed drastically across the range of our training data; 3-point attempts increased, and 2-point attempts decreased dramatically. We attempted to model this variability across different seasons using a feature called years_since, which is the integer number of years since 2001, but ultimately decided to exclude it because of the potential dangers of extrapolation. In a study involving the prediction accuracy of the death-rate and still-born rate using parabolas and straight lines, researchers showed that “both [extrapolated] representations have a mean error much in excess of the average mean error of interpolation” (Emily Perrin 1904). By including the years_since feature, all of the observations in our test data are now outside the range of our training data. Unlike other features in the test data, such as FGA, GP, and Assists, which are well within the respective ranges in the training data, years_since is the only feature whose test data values are strictly outside the range of the feature in the training data. Consistent with the research, we found that including years_since increased the testing error across all our classification models. Although we failed to do so, future researchers must implement a way to account for these differences, either through feature engineering or model tuning, to address these systematic differences.

Role Revolution

A common commentary on the modern age of basketball is how it has evolved into a "position-less" game. Sports pundits and academic researchers alike have suggested that the five original positions of basketball may no longer be best suited to describe the positions we see today. One paper in particular performed clustering to define 5 new positions to better capture the types of players we see today and a description of these positions can be found in Appendix section 5. This "role revolution" may be why some players were consistently misclassified across the multiple classifiers used in this project:



Misclassified Player	**Hypothesis for Misclassification**
Bojan Bogdanovic (SF/PF)	Although Bogdanovic is a forward, he is one of the best 3-point shooters in the league at the moment which is uncharacteristic for a forward.
Patrick Beverly (SG)	Beverly is a defense-oriented shooting guard with abnormally high defensive statistics including rebounds, blocks, and steals which likely led to him being classified as a forward.
Ben Simmons (PG)	Unlike most point guards, Simmons attempted zero 3-pointers in the 2022-2023 season. Simmons is known to be a poor shooter and as a result, his scoring statistics look a lot more like that of a forward than a guard.
Nikola Jokić (C)	Whereas traditional basketball centers do not make very many assists, Jokić has some of the highest assist numbers in the league across numerous seasons.

Figure 10: Misclassified Players

As noted in another paper, it is entirely possible that the position we have predicted for players is accurate and the opposite is true—perhaps the player is capable of playing in a different position (Frederico Bianchi 2017). Paul George of the Los Angeles Clippers for example is listed as a Forward in the 2022-2023 season but started his career as a Shooting Guard for his first two seasons in the NBA.(Bennett 1995)

References

Bennett, K. P. (1995), *Global tree optimization: A non-greedy decision tree algorithm*, *Rensselaer Polytechnic Institute*.

Emily Perrin (1904), "On some dangers of extrapolation," University College, London: Journal of Policy Analysis; Management.

Frederico Bianchi, P. Z., Tullio Facchinetti (2017), *Role revolution: Towards a new meaning of positions in basketball*, *Electric Journal of Applied Statistical Analysis*.

Fumitake Takahashi, S. A. (2002), *Decision-tree-based multiclass support vector machines*, *Institute of Electrical and Electronics Engineers*.

Jaime Sampaio, S. I., Manuel Janeira (2006), *Discriminant analysis of game-related statistics between basketball guards, forwards and centres in three professional leagues*, *European Journal of Sport Science*.

K. P. Bennett, J. A. B. (1998), *A support vector machine approach to decision trees*, *Institute of Electrical and Electronics Engineers*.

Thomas Sadler, S. S. (2016), *The 2011-2021 NBA collective bargaining agreement: Asymmetric information, bargaining power and the principal agency problem*, *Managerial Finance*.

Yolanda Escalante, A. G.-H., Jose M Saavedra (2010), *Game-related statistics in basketball by player position and final game score differences in european basketball championship 2007*, *Fitness & Performance Journal*.