STAT 441 Project Proposal

Mark Chen, Reven Liu, Arvind Nagabhirava, Rain Zhao

2023-04-02

STAT 441 Project Proposal

Dataset: The data was taken from the ESPN NBA Player Stats website found here: 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 more games the player has played in that season.

A package to scrape the data, create graphs and other statistical measures exists here: https://github.com/UBC-MDS/rsketball. Unfortunately, the package seems to be outdated or not fully developed as we were unable to install and use the package in our versions of R.

Thus, we instead collected the data manually, retrieving **9593** of data with **23** features. The data by season was manually copied over from the website into separate tabs in excel, which was then aggregated with Excel's built-in VSTACK function. Checks concerning the total number of observations per season, as well as the totals for each feature value were verified. The features in the data are as follows:

- SEASON: The season
- POS: Position
- GP: Games Played
- MIN: Minutes Per Game
- PTS: Points Per Game
- FGM: Average Field Goals Made
- FGA: Average Field Goals Attempted
- FG%: Field Goal Percentage
- 3PM: Average 3-Point Field Goals Made
- 3PA: Average 3-Point Field Goals Attempted
- 3P%: 3-Point Field Goal Percentage
- FTM: Average Free Throws Made
- FTA: Average Free Throws Attempted
- FT%: Free Throw Percentage
- REB: Rebounds Per Game
- AST: Assists Per Game
- STL: Steals Per Game
- BLK: Blocks Per Game
- TO: Turnovers Per Game
- DD2: Double Double
- TD3: Triple Double

The string-typed SEASON feature was transformed to generate additional integer-typed features:

- year: The Season Starting Year
- years_since: The Number of Years Since 2001

Research Question:
Investigate whether a classifier comprised of a combination of SVM and Tree-based methods can outperform SVM models and tree-based models. We will evaluate these classifiers on a dataset of NBA players to see which approach best classifies a player's position as either a Guard (PG, SG, G) or a Forward (SF, C, F, PF).

Classification Methods

1. Support Vector Machines (SVM):
   We consider the SVM classifier using the linear, polynomial, and radial-basis function kernels. 5-fold cross validation is used with grid search to tune the slack parameter C, the kernel coefficient γ, and the class weight. γ is only tuned for the polynomial and RBF kernels.
2. Bagging:
   The Bagging classifier is used with the base estimator as the Decision Tree Classifier. 5-fold cross validation is used to tune the base estimator loss criterion, maximum depth, minimum samples to split, minimum samples for leaves, and max feature configuration. For Bagging, we instead employ Randomized Search, noting its superior runtime over the exhaustive Grid Search.
3. Hybrid Approach:
   A combination of tree-based methods with SVMs.

Inspiration:
One of the limitations of tree-based methods is that they fit very complex decision boundaries and don't do well on simple decision boundaries, particularly on those that are close to linear.

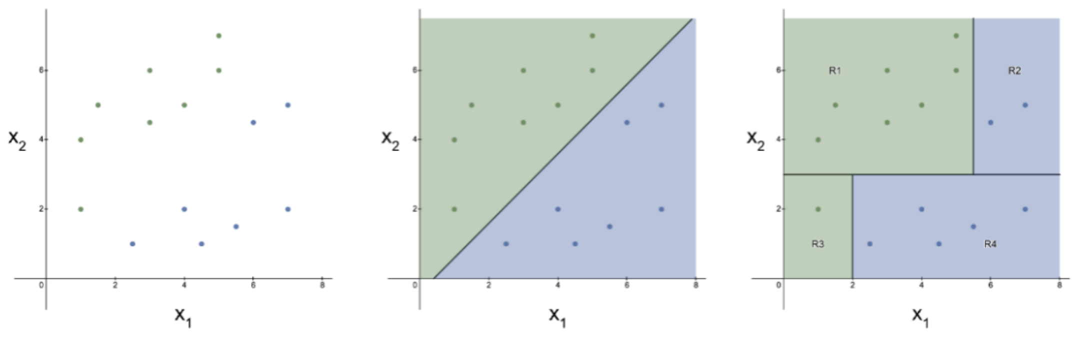

Figure 1: Left: Toy dataset. Center: SVM classifier applied to dataset with hyperplane $-x_1 + x_2 > -0.5$. Right: Tree based approach applied to dataset.

Algorithmic Sketch:
Build a tree in the following manner:
1. Use SVM to generate the optimal hyperplane.
2. Split the data based off the hyperplane from step 1.
3. For each subset of the data resulting from a split, split the subset in the same way, via another hyperplane using SVM.
4. Recursively split the data within each split using hyperplanes generated from running SVM on the subset of data within the given split until stopping criteria are met.

The key insight here is that the tree-based method requires far more splits to divide the dummy data in Figure 1. In our hybrid approach, we would allow the tree to make SVM-like splits, allowing the first split in the tree to be $-x_1 + x_2 > -0.5$, completely separating the data. For a more complicated dataset we would continue to make SVM splits as described in the algorithmic sketch until stopping criteria are met.

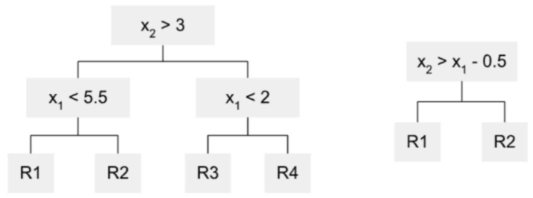

Figure 2: Left: The decision tree from a standard tree-based approach. Right: Decision tree from the hybrid approach where splits are hyperplanes.

Libraries:
- Scikit-learn for SVM and BaggingClassifier, calibration, and model evaluation
- Matplotlib for EDA and plots of feature importance
- Pandas and numpy for data manipulation and scientific computing

Preliminary Results

Preliminary analysis indicates SVM with RBF kernel performs marginally better than both the other SVM models and a Tree-based Bagging model. From the plot of permutation importance, it appears that rebounds and assists were two of the most important features in classifying player position. Conversely, features such as TD3 (triple double) appear to have minimal predictive power.

Model	Parameters	Error	Calibration Timing
SVM	Linear kernel	0.131	63s
SVM	RBF kernel	0.118	106s
SVM	Polynomial kernel	0.120	117s
Tree-based bagging	Decision tree base estimator	0.127	7s

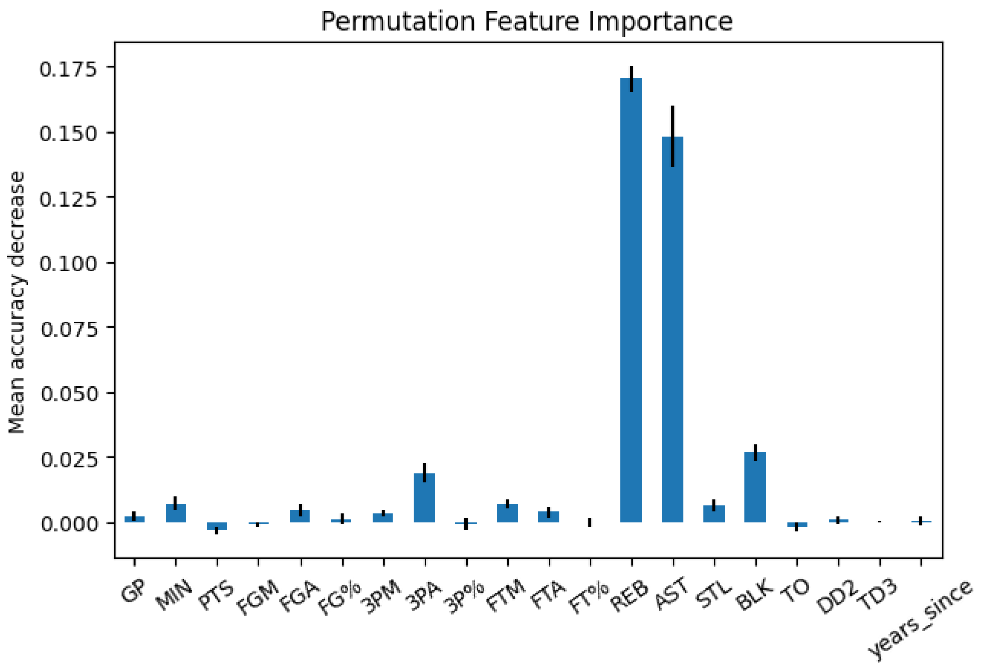

Figure 3: Permutation Feature Importance.