Discover Outstanding NBA Players: Comparing Anomaly Detection Techniques

Jason Chitla, Emily Jasienowski, Prajesh Singh, and James Ticatic
North Carolina State University
jachitla@ncsu.edu, eijasien@ncsu.edu, psingh8@ncsu.edu, jrticati@ncsu.edu

1 Background

1.1 Problem

Given key statistics for an NBA player, is he outstanding or not? That is the problem we are tackling. We want to train different anomaly detection machine learning models that identify outstanding NBA players from a group of players. We will then compare the performances of these models to see which techniques were more accurate in finding outliers (or players who were "outstanding"). Constructing the outlier detection models between all techniques follow the same stages of parameterization, training, and testing. Parameterization where you collect the raw data has already been done for us. We will be using NBA statistics data available from kaggle.com (more information in the data section). However, the training and testing stages is what our focus will be on for each technique. We will tune each model depending on each technique's hyper-parameters.

Our big focus is seeing how supervised anomaly detection compares to unsupervised anomaly detection. The supervised algorithms' training data will be labeled so the outliers are known. The unsupervised algorithms' training data will not be labeled but still contains outliers. The two supervised anomaly detection algorithms we will be using are: SVM and Supervised Robust Covariance Estimator (with Elliptic Envelopes). The two unsupervised anomaly detection algorithms we will be using are: Local Outlier Factor and a Decision Tree technique with Isolation Forests.

1.2 Literature Survey

Previous work has been done on NBA data using different outlier detection algorithms. Junming Shao, et al. [1], used Synchronization Based Outlier Detection to find outliers in NBA data from the 2008-2009 season. They only used 4 features for 444 rows (players): games played, points, rebounds, and assists. They had great results, finding 18 outliers consisting of some of the best players from that season (including Lebron James). Their method of detecting outliers based on synchronization was inspired by the phenomenon of synchronization. This phenomenon suggests that a group of events spontaneously find a common rhythm despite their differences. In more related work, Yuan Li, et al. [2], compares subspace-based methods of outlier detection to the popular distance-based and density-based methods. Subspace-based methods attempt to optimize outlier detection for high dimensions. It works like the kernel trick in SVM by examining projections with low dimensions. They used NBA data from nba.com and had 425 rows with 20 features. With that many features, they found the three best subspaces and the outlier detection worked great, picking out special players. In one more piece of related work, Cheng Fan, compares four methods: Local Outlier Factor (LOF), Angle-Based Outlier Degree (ABOD), Subspace Outlier Degree (SOD), and Random Subspace Projection (RSP) [3]. He obtained his data from databasebasketball.com. His data consists of 20 features and 441 rows. Cheng Fan structured his work in first detecting outliers and then finding the attributes that cause them to be classified as such.

We just surveyed existing work that has been done on our problem, but now let's look at existing work that has been done on our approach. Besides using purely distance-based approaches, we are using a model-based approach. Our supervised models first learn normal behavior from the

given data and then become capable to detect outliers. We then fit two unsupervised models. The first unsupervised model is with elliptic envelopes and the next one is with decision trees. Salima Omar, et al. [4], compared different model-based approaches to outlier detection in their research and found that their supervised based methods significantly outperformed their unsupervised based methods. SVM in particular was the best supervised technique. Other supervised algorithms they analyzed were: neural networks, k-Nearest Neighbors, Bayesian Networks, and Decision Trees. The unsupervised algorithms they analyzed were: K-means, Self-organizing maps (SOM), C-means, Expectation-Maximization Meta Algorithm (EM), Adaptive resonance theory (ART), unsupervised Niche Clustering (UNC), and One-Class SVM. They listed pros and cons for each of these techniques. Pros they found for Decision Trees were that they are simple to understand and interpret, they are robust, and they perform well with large data in a short amount of time. Cons they found for Decision Trees were that sometimes they create overly complex trees that do not generalize the data well. Pros they found for SVM is that they can deal with very high dimensional data and they usually work very well. However, a con they listed for SVM was that it requires a lot of memory and CPU time. Zeeshan Ahmad Lodhia, et al. [5], also analyzed popular machine learning and outlier detection techniques. However, besides just reviewing supervised and unsupervised techniques, they also analyzed reinforcement learning techniques as well. Reinforcement learning is where a model is continually corrected after making an incorrect classification. They analyzed Markov Decision Process and Monte Carlo approximation. Markov decision process is a model for decision making where outcomes are somewhat random. They are used in reinforcement learning because the probability of these different outcomes can be tuned. Monte Carlo method is where you repeatedly execute random sampling to obtain results. The key here is using randomness to solve problems that are deterministic. Monte Carlo methods learn directly from experience.

2 Method

In this section, we describe the intelligence behind the various supervised and unsupervised anomaly detection algorithms we use to identify outstanding NBA players.

2.1 Support Vector Machine (SVM)

In order to describe how we utilized SVM to identify outstanding NBA players, it is important to understand what an SVM is and how it works. SVM is a supervised learning model that assigns data to one of two classes based upon the input features. SVM determines the optimal hyper-plane that separates the two classes by the maximum distance. This selection of hyper-planes is an optimization problem. If the data is linearly separable then linear SVM should be used. However, if the data is not linearly separable, mapping the data to a higher dimensional space makes SVMs work very well. The original feature space can always be mapped to some higher-dimensional feature space where the training set is guaranteed to be separable. This is called the "kernel" trick. Different kernel functions can be utilized to create different high-dimensional feature spaces. Some common kernel functions are linear, polynomial, radial basis kernel, and sigmoid.

There are hard-margin SVMs and soft-margin SVMs. Hard-margin SVMs are models where all data points are on the correct side of the separating hyper-plane. Soft-margin SVMs are models where there exists some flexibility on crossing this plane. In order for this to happen, a C value, or regularization parameter, is introduced. As this regularization parameter increases, the number of support vectors decreases. This is because C is the penalty on the slack variables and if you increase it, the greater the penalty is put on violating constraints. Therefore, the solution changes to reduce these violations by making the margin narrower and thereby leading to fewer support vectors. Another parameter used to tune the performance of SVMs is the gamma value. The gamma value defines how far the influence of a single training example reaches. Low gamma values mean points far away from the hyper-plane will have influence in classification and high gamma values will result in low influence from far away points. At a low-level, SVMs solve the dual problem to obtain support vectors, α_i. The following formula is used:

$$L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j yyk(x^i, x^j)$$

Then the derivative of L is taken with respect to α_i for each i. System of equations can then be used on the resulting equations to find each alpha value. Then w and w_0 can be found from the following formulas:

$$w = \sum_{i \epsilon SV} \alpha_i y_i \phi(x^i)$$

$$w \cdot \phi(x) + w_0 = 0$$

Finally, at classification time, the following computation is used to identify the class for each new data point:

$$sign(w \cdot \phi(x) + w_0 = 0)$$

2.2 Supervised Robust Co-variance Estimator with Elliptic Envelope

Co-variance estimation is an estimation of the shape of the data's scatter plot. Data sets are regularly littered with outliers. Empirical and shrunk covariance estimators are sensitive to outliers [6]. This is where robust co-variance estimators become very useful. They downweight or even discard some data points if they deem them to be outliers. Since Elliptic Envelopes are a form of robust co-variance estimation, they are not influenced by outliers.

For supervised robust co-variance estimator with elliptic envelopes, we are able to fit the tightest ellipse possible while discarding some "contamination" points. Since the shape of the ellipse is made with no influence from the outliers, the outliers can then be detected outside the ellipse. Contamination is a parameter to tune. It is an estimation of the proportion of outliers in the data set. In order to find the best contamination value, we will calculate the actual contamination proportion (since we have the ground truth labels).

2.3 Local Outlier Factor

Local Outlier Factor is an unsupervised anomaly detection method which is not model-based like our other approaches. This technique is a density-based approach. It computes the local density deviation of all data points with respect to its neighbors [7]. Each data point has an anomaly score. The score depends on how isolated the point is based off of the density deviation. The outliers are the data points that have a lower density than their neighbors. The number of neighbors to compare each point to is a parameter you can tune. Usually, this parameter is set to 20 because that works well in general. Like elliptic envelopes, contamination is also a parameter you can tune.

The following formulas are what Local Outlier Factor utilizes to compute the anomaly score for each data point [8]. The first formula calculates the local reachability distance for an object o:

$$lrd_k(o) = \frac{||N_k(o)||}{\sum_{o'} \epsilon N_k(o)^{reachdist_k(o,o')}}$$

This second formula compares a given object's density with its neighbors to find the degree that the object is an outlier:

$$LOF_k(o) = \frac{\sum_{o'} \epsilon N_k(o) \frac{lrd_k(o')}{lrd_k(o)}}{||N_k(o)||} = \sum_{o' \epsilon N_k(o)} lrd_k(o') * \sum_{o' \epsilon N_k(o)} reachdist_k(o' \leftarrow o)$$

The resulting value is the average of the ratio of the local reachability density of the object o and those of o's k nearest neighbors.

2.4 Decision Trees With Isolation Forests

Decision trees with isolation trees is one of our unsupervised outlier detection techniques. Isolation forests are a great way to detect outliers with unlabeled data. They are built with decision trees.

Partitions are created on these trees by first selecting a feature and then randomly splitting between the min and max of that feature. The number of splits required to isolate a sample is the same as the length from root node of the tree to actual node [9]. This length is a measure of normality. This length is shorter for anomalies so when a forest of trees produce short lengths for certain data points, they are most likely outliers.

At a low-level, an anomaly score is calculated for each data point so that a decision can be made on whether or not that data point is an outlier. Scores can range from 0 to 1, where scores closer to 1 suggest an outlier. If all scores hover around 0.5, then there are no outliers in the data.

3 Plan and Experiment

3.1 The Question

Our main question we are answering is whether supervised or unsupervised anomaly detection techniques work best. While it may seem obvious that supervised anomaly detection techniques should perform better, we reasoned that maybe an unsupervised technique will find a relationship between the features that we don't see.

3.2 Data

We used NBA and ABA statistics data for the 2004-2005 season. This dataset is free from kaggle.com. It contains players' regular season stats, regular season career totals, playoff stats, playoff career totals, allstar game stats, team regular season stats, complete draft history, NBA coaching records by season, and NBA career coaching records. We only needed to use the players' regular season career data. This included each player's: name, league played in, games played, minutes played, total points scored, offensive rebounds, defensive rebounds, total rebounds, assists, steals, blocks, turnovers, personal fouls received, field goals attempted, field goals made, free throws attempted, free throws made, three pointers attempted, and free throws made.

Clean the Data

In order to get the best performance out of all the anomaly detection techniques we are using, we had to clean our data. According to the *"Garbage In, Garbage Out"* principle, if our data is bad, our models will be bad too.

The first step we performed to clean and organize our data was removing three unnecessary columns from the excel sheet itself: league played in, amount of offensive rebounds, and amount of defensive rebounds. The league played in specified whether a player played in the NBA or the ABA. We determined this did not correlate to whether a player could be classified as outstanding or not. The next two columns we removed specified how many offensive and defensive rebounds a player had. These columns are supposed to sum up to the total rebounds column but that was not always the case in our data. This discrepancy could cause confusion for our models in determining each feature's relation to the dependent variable (outstanding or not). We decided total rebounds per player did a good enough job and it was unnecessary to break them up between offensive and defensive. Determining efficiency for each player based on position (offensive or defensive) is not an objective we have so we dropped these columns.

The next step in cleaning our data was to get rid of low-quality data points. This is easier to do with pandas rather than in the excel sheet itself. After importing the csv file into our python program, we deleted all data points that had a value of less than 500 in the "minutes played" column. Most of the time, if a data point had less than 500 minutes played it was a zero. These zeros could mess up our model's by giving this column an improper weight and not show the correct relationship between the other input features. However, if the value was not zero but still less than 500, we still removed the sample. We did this for the same reason that we removed all data points that had a value of less than 100 in the "games played" column. This is not because the data is simply not there like for the data points with a value of zero in "minutes played." Rather, this is because we decided players who have not played at least 100 games should not be included. They are still defining who they are in the basketball league. It is unfair and unrepresentative of whether a player is outstanding or not by looking at data that is the very beginning of their career.

Steals, blocks and turnovers were not official NBA statistics until the 1970's. Therefore, those statistics are listed as zero during those times. Our data includes some players' careers which start before the 1970's (and some even ending before the 1970's). This is a problem. In order to fix this problem, we removed all data points that had a value of zero for steals before 1970. Even though steals, blocks, and turnovers were not official NBA statistics before 1970's some players during that time period still had those statistics and we wanted to make sure to include those players. The starting season for each player was another available column apart of our data. However, it was not apart of the players' regular season career csv file. It was in a different csv file. Since each data point had an id, we were able to transfer this column over.

After cleaning up our data, we are left with 1504 rows and 15 features. 6.65% of the 1504 rows are outstanding players while the rest are non-outstanding, average players. In a second, you will see the formula we used to identify outstanding players in our data.

Generate Ground Truth Labels

Supervised anomaly detection algorithms require labeled data. This means for every data point we need a column which specifies whether it is a normal or abnormal point (by being assigned a 1 or a 0). Our data set does not specify this out of the box. We must identify a way to classify each data point as an outlier or not.

What makes a basketball player outstanding? What kind of metric do you use? We decided to use player efficiency rating to rank each player. The formula is:

$$(PTS + REB + AST + STL + BLK - MissedFG - MissedFT - TO)/GP$$

Our ratings ranged from 0.88 to 41.50. We decided to classify the top 100 players ranked based on this efficiency rating as outstanding.

3.3 Steps

After cleaning the data and calculating the ground truth labels, we moved on to fitting the models for each anomaly detection technique. First, we created our two models with our two supervised anomaly detection algorithms.

To find the best SVM model, we created 4 SVM models each with a different kernel: radial basis kernel, linear, polynomial, and sigmoid. To train each model, we used stratified random sampling to split up our data into train and test sets. We used the train_test_split function in sklearn to make sure to get the same proportion of outliers in both train and test sets. We decided to make the test set 30% of the 1504 total rows of data. We then went further to split up the test set into an outstanding player test set and a non-outstanding player test set. This is because if we just used the one test set, we wouldn't know how well the model was at predicting the outliers specifically. For example, the accuracy could be 97%, but that would not tell you how many outliers it was successful at classifying. Then, to find the best parameters we used GridSearchCV. GridSearchCV tries each combination of regularization and gamma values passed in. We passed in $[0.001, 0.003, 0.007, 0.01]$ regularization values to try and $[0.001, 0.003, 0.007, 0.01]$ gamma values to try. Then we had each model make predictions and compared them to the ground truth values for each test set.

Next, we trained the supervised elliptic envelope. We recast our y values to -1 and 1 since Elliptic Envelopes utilizes these values. This is a small implementation detail. These values represent our two classes. Since we seeded the random generation of the creation of the test sets and the training of the models, we know for a fact that the actual proportion of outliers in the training data is 6.7% but we try a couple other values to see how it affects our model's accuracies. First, we create a general table showing all the model's performances with a default contamination value for all the models (not SVM). Then we go more in-depth, and show how tuning the parameters affects each model's performance. We do this after creating the general table. It is important that we continue to test with the same test sets we generated earlier, so we can see how the models stack up against the same test sets.

To finish the general table to show all models' performance across default parameters that we select, we now train our two unsupervised anomaly detection models. First, we trained our Local Outlier Factor model. We had to set the novelty parameter to "True". The Local Outlier Factor model object will not allow us to make predictions if you leave it set to "False". Next, we want to find the best

combination of the k value and the contamination value. We tried $[500, 600, 700, 1000]$ k values with $[0.05, 0.067, 0.1, 0.2]$ contamination values and discovered a wide range of accuracies. Finally, we train our unsupervised isolation forest model. There is nothing special to do here. The sklearn $IsolationForest$ object does all the magic.

Lastly, we loop through these different contamination values: $[0.1, 0.2, 0.3, 0.4]$ and compare them across Elliptic Envelope, Local Outlier Factor, and Isolation Forest models. We generate a table to easily view how these contamination values affect these models differently.

4 Results

Our results for the general table are given below:

	C	Gamma	Normal Accuracy (%)	Outlier Accuracy (%)	Contamination
SVM - RBF	0.001	0.001	100.000000	0.000000	N/a
SVM - Linear	0.003	0.001	100.000000	90.000000	N/a
SVM - Poly	0.001	0.001	98.104265	96.666667	N/a
SVM - Sigmoid	0.001	0.001	100.000000	0.000000	N/a
Supervised Elliptic Envelope	N/a	N/a	95.971564	66.666667	0.0665399
Local Outlier Factor	N/a	N/a	96.445498	66.666667	0.0665399
Isolation Forests	N/a	N/a	96.445498	53.333333	0.0665399

Normal accuracy is the accuracy of correctly identifying non-outstanding players. Outlier accuracy is the accuracy of correctly identifying outstanding players.

Notice that SVMs with RBF or sigmoid kernels have 0% outlier accuracy whereas SVMs with linear or polynomial kernels have 90% and 97% outlier accuracies respectively. This strongly supports that our data is linearly separable.

In this next table, you can see how different combinations of number of neighbors, k, values and contamination values, c, affect the Local Outlier Factor model.

	k	C	Normal Accuracy (%)	Outlier Accuracy (%)
0	500	0.050	96.919431	53.333333
1	500	0.067	96.445498	66.666667
2	500	0.100	95.734597	73.333333
3	500	0.200	88.388626	80.000000
4	600	0.050	96.919431	53.333333
5	600	0.067	96.445498	66.666667
6	600	0.100	95.734597	73.333333
7	600	0.200	88.151659	80.000000
8	700	0.050	96.919431	53.333333
9	700	0.067	96.445498	66.666667
10	700	0.100	95.734597	73.333333
11	700	0.200	88.151659	80.000000
12	1000	0.050	96.208531	53.333333
13	1000	0.067	96.208531	53.333333
14	1000	0.100	95.260664	66.666667
15	1000	0.200	87.440758	80.000000

The best combination is a k value of 500 with a contamination value of 0.2. With those parameters you get a normal accuracy of 88.39% and an outlier accuracy of 80%. That is not the best normal accuracy, but that is the expense of a better outlier accuracy. We can see that the contamination value is the parameter that has more of an effect on the model. The range of each outlier accuracy for each k value is 53.33% to 80.0% and the range of each normal accuracy for each k value is about 87-88% to 96-97%. Increasing the contamination value decreases normal accuracy and increases outlier accuracy. This is observed in our last table as well.

In this last table, you can see how the different contamination values affect Elliptic Envelope, Local Outlier Factor, and Isolation Forest models differently.

	Normal Accuracy (%)	Outlier Accuracy (%)	Contamination
Elliptic Envelope	94.786730	73.333333	0.1
Local Outlier Factor	95.734597	73.333333	0.1
Isolation Forests	95.497630	63.333333	0.1
Elliptic Envelope	89.336493	90.000000	0.2
Local Outlier Factor	88.388626	80.000000	0.2
Isolation Forests	86.966825	83.333333	0.2
Elliptic Envelope	77.251185	96.666667	0.3
Local Outlier Factor	79.146919	90.000000	0.3
Isolation Forests	76.303318	93.333333	0.3
Elliptic Envelope	68.009479	96.666667	0.4
Local Outlier Factor	66.587678	96.666667	0.4
Isolation Forests	68.009479	100.000000	0.4

There is a larger increase in outlier accuracy for Local Outlier Factor, when increasing the contamination value, compared to the other models. This suggests that Local Outlier Factor is more sensitive to contamination value than the other models. The normal accuracy drops roughly similar across all models as the contamination value is increased. The contamination value is the estimated proportion of outliers in the data. When you increase this proportion, this lets the models know that they should be finding more outliers.

In the end, our main question was answered: "Do supervised or unsupervised anomaly detection techniques work better?" Our results strongly support that supervised anomaly detection techniques work better. SVM with a linear kernel obtained a normal accuracy of 100.0% with an outlier accuracy of 90.0% while SVM with a 3-degree polynomial kernel obtained a normal accuracy of 98.1% with an outlier accuracy of 96.7%. None of the other models came close to those combination of accuracies.

5 Conclusion

In conclusion, we found an answer to our main question of whether supervised or unsupervised anomaly detection techniques work best. The supervised techniques outperformed the unsupervised techniques. We also learned a lot. In the end, the data tells the story. We saw how important it was to pre-process our data and clean it, so the models can make the most out of the data. In regards to the problem itself, of detecting outstanding players in basketball data, defining the formula for "outstanding" is really important. We used the NBA player efficiency formula, but there are many other formulas to use. This formula is what directly taught our supervised models and is what graded our unsupervised models. What if our unsupervised models discovered a new formula to grade NBA players by? We also learned that tuning models' parameters is critical to obtaining the best results. If we had more time, we would conduct principal component analysis (PCA) on each model to see which features had the largest affect on whether a player is outstanding or not. However, you can build a solid understanding of this by just looking at the NBA player efficiency formula (which we used to calculate our ground truth labels). We are more than satisfied with our final results and are grateful for working on such an interesting project.

References

[1] Shao, J., Böhm, C., Yang, Q., & Plant, C. Synchronization Based Outlier Detection. Retrieved December 11, 2018, from http://dm.uestc.edu.cn/wp-content/uploads/paper/Synchronization-based-outlier-detection.pdf

[2] Li, Y., & Kitagawa, H. A Robust Method for Detecting DB-Outliers from High Dimensional Datasets. Retrieved December 11, 2018, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.492.3924&rep=rep1&type=pdf

[3] Fan, C. Using Outlier Detection Algorithms to Analyze NBA Players. Retrieved December 11, 2018, from https://cran.r-project.org/web/packages/HighDimOut/vignettes/GoldenStateWarriors.html

[4] Omar, S., Ngadi, A., & Jebur, H. H. Machine Learning Techniques for Anomaly Detection: An Overview. Retrieved December 11, 2018, from https://pdfs.semanticscholar.org/0278/

[5] Lodhia, Z. A., Rasool, A., & Hajela, G. A survey on machine learning and outlier detection techniques. Retrieved December 11, 2018, from http://paper.ijcsns.org/07_book/201705/20170536.pdf

[6] Scikit-Learn. Covariance Estimation. Retrieved December 11, 2018, from https://scikit-learn.org/stable/modules/covariance.html

[7] Scikit-Learn. Outlier detection with Local Outlier Factor (LOF). Retrieved December 11, 2018, from https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html

[8] Chepenko, D. A Density-based Algorithm For Outlier Detection. Retrieved December 11, 2018, from https://towardsdatascience.com/density-based-algorithm-for-outlier-detection-8f278d2f7983

[9] Scikit-Learn. Novelty and Outlier Detection. Retrieved from https://scikit-learn.org/stable/modules/outlier_detection.html

Github Code: https://github.ncsu.edu/jachitla/CSC422_P04_nba_outliers/tree/master