NBA On-Ball Screens: Automatic Identification and Analysis of Basketball Plays

Andrew Yu
Department of Computer Science and Software Engineering
Penn State Erie, The Behrend College
Cleveland, OH
andrewyu@psu.edu

Dr. Sunnie Chung
Department of Electrical Engineering and Computer Science
Cleveland State University
Cleveland, OH
s.chung@csuohio.edu

***Abstract* — In basketball, the on-ball screen is a dynamic offensive strategy that involves the movement of multiple players and the ball to create an effective shot attempt. It is a fundamental play employed by all teams in the National Basketball Association (NBA), the highest level of competition for basketball. The analysis of a basketball team and its strategies must first include the identification of such plays. With the presence of sophisticated data collection and analysis tools, this paper proposes methodologies to extract, transform, and analyze player motion-tracking data in NBA games to automatically identify the presence of on-ball screens, with 90% sensitivity, an 8% improvement on existing literature.**

***Keywords* —** *Data analytics, sports analytics, basketball, NBA*

I. INTRODUCTION

Sports analytics is the management and analysis of data collected from sports games to quantify past results and predict future outcomes. Since the rise of professional sports in the 19th century [1] and its commercialization in recent decades, sports analytics has grown from simple box scores and team managers to multi-million dollar analytics teams equipped with state-of-the-art technologies.

A. Background

In the National Basketball Association (NBA), the home team of each game employs a group of scorekeepers who record basic statistics like points, rebounds, and assists. Such statistics are easily accessible and considered common knowledge to all teams, players and fans. Teams looking for more insight from the game hire their own analysts who are often assigned the grueling task of watching video recordings of basketball games to manually collect data.

In 2012, the development of sophisticated tools for collecting player motion-tracking data was introduced by Stats LLC, named STATS SportVU® Basketball Data (henceforth called "SportVU data"). Since then, NBA teams have been analyzing the new influx of basketball data. Because of the competitive nature of professional sports, analytics departments of NBA teams operate behind the scenes, rarely if ever divulging their findings to the public; but when playoffs start and games become more important, every decision on the basketball court is more deliberate, and the influence of analytics shines through.

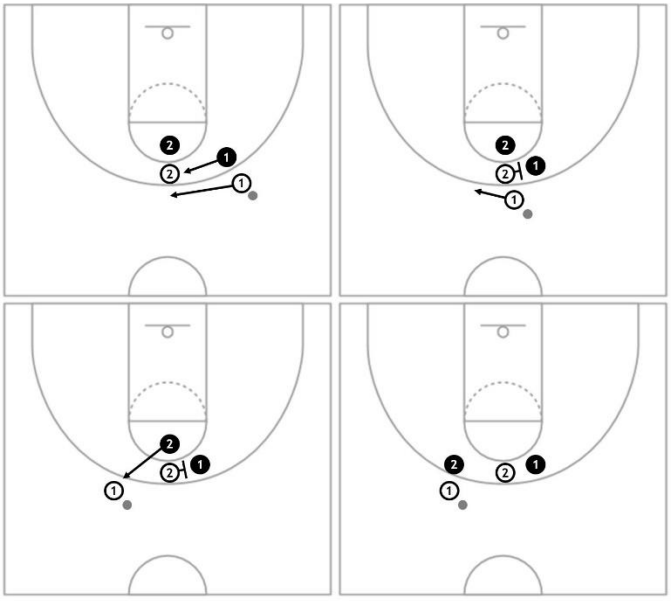

Fig. 1. The progression of an on-ball screen

B. On-Ball Screens

The on-ball screen involves four players, two on defense guarding two on offense, one of which is holding the ball.

First, the offensive player without the ball (Fig. 1, white 2), called the screener, stands still and "sets a screen", effectively making himself an obstacle for anyone trying to move through his area (Fig. 1, top left). Then, the offensive player with the ball (white 1), called the ball-handler, approaches and brushes by the screener. The defensive player trying to stay close to the ball-handler (black 1), called the ball defender, will have his path blocked by the screener, putting him in a poor position to defend the ball-handler (top right). The positional advantage produced by the screen on the ball is the purpose of the on-ball screen.

The defensive team may have its own strategy to mitigate the effects of the on-ball screen. For example, while the ball defender struggles to recover by going around the screen, another defender (black 2) may step up to defend the ball-handler (bottom left). In the end, a different defender is guarding the ball-handler and a "switch" has occurred (bottom right). The switch refers to the fact that two defenders have switched places in guarding their opponents.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

The offensive team may expect the switch and exploit its outcome for their advantage. Often, the goal of the on-ball screen is to force a poor defender to defend a good offensive player.

There are countless other defensive reactions and offensive counter-reactions that may arise, but they all stem from the same beginning: the on-ball screen. Therefore, their identification is the first step in predicting how an opponent will run and defend the on-ball screen, and then generating plays that exploit the weaknesses of the opponent's tendencies.

Because on-ball screens are highly dynamic actions involving multiple players and various lengths of time, identification of on-ball screens is typically done manually in painstaking fashion by watching video recordings of the game. A human observer with a modicum of basketball knowledge can readily identify on-ball screens, but it is a time-consuming process. Because of this, teams may resort to watching only a few most recent games that the upcoming opponent has played, to approximate its current strategies in employing and defending against the on-ball screen. This process is prone to human error and bias, and reduces the quality of the subsequent analysis. The ability to automatically identify on-ball screens would dramatically reduce the time spent and improve the subsequent analysis.

II. Related Works

McQueen et al. [4] unveiled a predictive algorithm for on-ball screens that achieved a sensitivity of 82% and positive predictive value of 80%. However, it suffered from a lack of available data. McIntyre et al. [3] followed up using a more robust dataset for a deeper look at the identification of on-ball screens, subdividing them into multiple categories based on the defense's reaction to the screen. Based on these findings, the study could produce a simple defensive strategy for a theoretical scenario, involving different offensive and defensive players.

The above papers offer a heavily limited view of the processes involved in developing a predictive classifier for basketball plays. To rebuild or improve upon them, they must first be recreated from scratch.

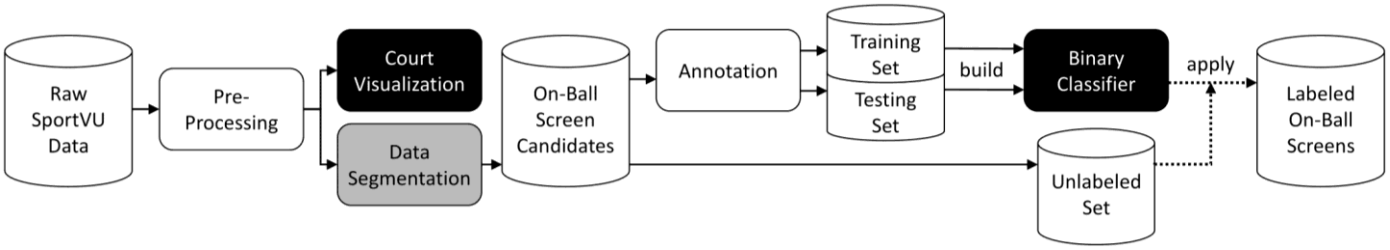

Fig. 2. A simplified data flow pipeline for the Framework

III. Data Gathering and Preprocessing

Data from three different sources on NBA games are collected to enable the automatic identification of on-ball screens:

- SportVU player motion-tracking data (SportVU data): SportVU is a proprietary technology from Stats LLC in partnership with the NBA since 2012 to provide full-featured motion-tracking capabilities and data feeds to all 30 NBA arenas [7]. SportVU utilizes a six-camera system installed in basketball arenas to track the real-time positions of all 10 players and the ball 25 times per second [5].
- Play-by-plays: Information such as scoring plays, rebounds, turnovers, fouls, timeouts, and substitutions are collected by scorekeepers in each NBA game. The play-by-play tables are necessary to build higher-level basketball concepts, such as possessions.
- NBA game tape: Television broadcast video of the basketball games must be used clarify the SportVU data when the SportVU technology produces errors and artifacts. It can also be used to build a labeled set for building a system capable of automatically identifying on-ball screens.

A. SportVU Data

NBA's official in-depth stats website was the source of the SportVU data [6]. In addition to general NBA statistics like box scores, the website provides advanced statistics that are derived using player motion-tracking data, including number of dribbles, average defender distance, average pass speed.

In order to calculate such figures, the client web browser sends an Asynchronous JavaScript and XML (AJAX) request for and receives a subset of the SportVU data for a single event in JavaScript Object Notation (JSON) format, which usually corresponds to a single possession in an NBA game. AJAX requests are used by the web application to dynamically request and receive data without having to reload the page. The web application typically requests and receives the SportVU data, calculates the relevant advanced statistics, displays them on the client web browser, then discards it. Instead of discarding the data after each session, they can be collected and compiled.

Looking inside one of the JSON files, a consistent hierarchical tree structure is found [large image]. Nodes at the top of the tree contain information identifying to which game and relative time frame in game the file belongs.

Nodes at the bottom contain the main data of interest: Cartesian coordinates of players and basketball on the court accompanied by timestamps. Each event JSON file contains about six seconds of coordinate data on average. In reality, the duration of time for each file varies wildly. Some files have no

coordinate data at all; they are empty shells without any useful data, apparently an artifact of the way SportVU data is recorded. For the rest, the SportVU player motion-tracking system records the 2-dimensional Cartesian coordinates of ten players and the basketball on the court in an approximation to 1/100,000th of a U.S. foot and a player ID string for each player 25 times per second. The radius of the ball from the perspective of a camera above is also recorded to simulate the vertical axis, or height, of the ball. Relevant time information, such as timestamp, game clock time, and period, are recorded for each frame as well.

B. Play-By-Plays

Basketball-Reference is a third-party provider of NBA statistics on the internet, which publishes play-by-plays in a convenient Hypertext Markup Language (HTML) table format [9]. Play-by-play tables are used to build higher-level basketball concepts, namely possession and directionality. Possession refers to the team who is controlling the ball. Directionality refers to the hoop on which each team is trying to score. For ease of understanding, one hoop is always referred to as the right hoop, other the left hoop. Directionality changes at half time, or after half of the game has been played.

Unfortunately, play-by-plays do not contain information about possession and directionality themselves; they must be derived using other information available in play-by-plays and the SportVU data. First, Possession must be determined for every moment in the game; then, obtaining directionality is relatively simple.

1) Possessions

Many entries in the play-by-play (events) denote a change of possession. Each event has an accompanying description. Each event is parsed to produce the current score, the main outcome of the event, time information, and the players involved if applicable. Using the players involved, possession data for that event can be determined.

For some events, possession is not clear for one or both time periods before and after the event; if an event describes a player missing a 2-point basket, it implies that the offensive player's team had possession from the last event to this event, but does not indicate which team will possess the ball after; that depends on which team rebounds the ball. Jump balls are the worst in this sense; it does not indicate who had possession prior to or after it.

To resolve this issue, a second pass iterates through each event backwards after assigning it a possession designation. If an event's predecessor has an unknown possession after, then the current event's possession prior is assigned to the predecessor's possession after, to fill in the gaps. If an event has an unknown possession prior and its predecessor has an unknown possession after, then there is a true gap between those two events whose possession cannot be determined using play-by-play data. However, those events, namely dead ball time outs, jump balls, and player substitutions, occur with no game clock elapsing between them, meaning that such gaps can be ignored without cost.

TABLE I. PLAY-BY-PLAY POSSESSION PROPAGATION

Event by Team 1	Possession Prior	Possession After
Turnover	Team 1	Team 2
Make 2-pt Shot		
Make 3-pt Shot		
Make Free Throw 2 of 2		
Make Free Throw 3 of 3		
Offensive Foul		
Personal (Defense) Foul	Team 2	Team 2
Personal (Defense) Take Foul		
Personal (Defense) Block Foul		
Loose Ball Foul		
Offensive Rebound	Team 1	Team 1
Make/Miss Free Throw 1 of 2		
Make/Miss Free Throw 1 of 3		
Make/Miss Free Throw 2 of 3		
Defensive Rebound	Team 2	Team 1
Shooting Foul		
Miss Free Throw 2 of 2	Team 1	?
Make Free Throw 3 of 3		
Miss 2-pt Shot	Team 1	?
Miss 3-pt Shot		
Time Out (Live Ball)	Team 1	Team 1
Time Out (Dead Ball)	?	Team 1
Jump Ball	?	?
Player Substitution		

2) Directionality

Directionality depends on the team and the period. The direction of play is determined by the visiting team for the first half, then swapped for the remainder of the game [10]. However, this information is not available to the public, and directionality depends on the observer's perspective. To avoid confusion, only the play-by-play and the Cartesian coordinates of the SportVU data is used to determine directionality.

. When the first field goals of the first and second periods are made, the play-by-play events and the coordinates of the basketball are recorded, noting which team scored the baskets, and on which side the ball is. This information is sufficient to determine the direction of play of either team in any quarter.

C. NBA Game Tape

Television broadcast video of the basketball games is used in this thesis to help create a labeled set of on-ball screens. They are necessary to clarify the SportVU data because Cartesian coordinates, even when visualized, often do not provide enough information to determine if an action is a labeled play or not. A player's stance, hand, leg, and body positioning, and other contextual information is destroyed when SportVU technology converts humans into points on a Cartesian grid. Local and national television broadcast recordings are used.

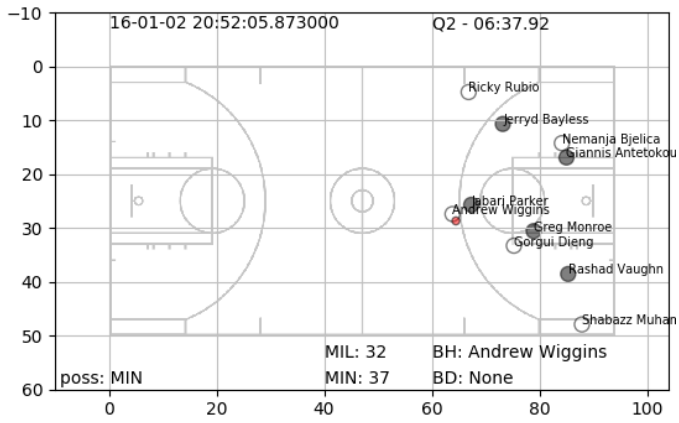

Fig. 3. A visualization of the basketball court and its players in an NBA game

IV. Data Segmentation

find_ball-handler(single frame):
1: ball-handler ← argmin(distance between ball and players on offense)
2: **if** the distance between ball-handler and basketball > 5 ft.
3: **then return** NULL
4: **else return** ball-handler

Fig. 4. Algorithm for finding the ball-handler

find_ball-defender(single frame, ball-handler):
1: **if** ball-handler does not exist **then return** NULL
2: ball-defender ← argmin(distance between ball-handler and players on defense)
3: **if** the distance between ball-handler and ball-defender > 12 ft.
4: **then return** NULL
5: **else return** ball-defender

Fig. 5. Algorithm for finding the ball-defender

find_screener(single frame, ball-handler):
1: **if** ball-handler does not exist **then return** NULL
2: screener ← argmin(distance between ball-handler and players on offense)
3: **if** the distance between ball-handler and screener > 10 ft.
4: **then return** NULL
5: **else return** screener

Fig. 6. Algorithm for finding the screener

find_action(13 or more consecutive frames in chronological order):
1: ball-handler ← find_ball-handler(first frame)
2: **if** ball-handler does not exist **then return** NULL
3: ball-defender ← find_ball-defender(first frame)
4: **if** ball-defender does not exist **then return** NULL
5: screener ← find_ball-screener(first frame)
6: **if** screener does not exist **then return** NULL
7: **for each** frame
8: **if not** find_ball-handler(frame) = ball-handler
9: **or** find_ball-defender(frame) = ball-defender
10: **or** find_screener(frame) = screener
11: **or** screener is not inside the paint
12: **or** basketball is not inside the paint
13: **or** argmin(distance between basket and screener) < 2 ft.
14: **then return** NULL
15: **return** frames

Fig. 7. Algorithm for extracting actions, or candidates for on-ball screens

Around 95 on-ball screens occur in a typical NBA game, and each on-ball screen occurs within six seconds of gameplay, usually lasting about 1.3 seconds. This means that a typical game contains just over two minutes of on-ball screen-relevant frames, a small fraction of total frames available. Therefore, it is wasteful to consider every frame using an intelligent algorithm to determine if an on-ball screen occurs or not; instead, a rule-based algorithm is devised to slice the data into segments (actions).

The rule-based algorithm is inclusive; it is intentionally designed to err on the side of accepting too many series of frames as actions rather than too few, because it is a first-pass filter. Any on-ball screens that are missed by the rule-based algorithm are not considered in the following layers, which may be costly for the overall performance.

V. Annotation

To improve upon the inclusive rule-based algorithm, a more selective intelligent algorithm must be employed. The intelligent algorithm is used to solve a classic binary classification problem: given an action, determine whether it is an on-ball screen or not. A supervised classifier is used, which requires a sample of labeled actions.

To label an action, the corresponding segment of NBA game tape and animated visualization of the court is studied by the human eye to determine whether it was an on-ball screen or not. This is a process that is prone to human error and bias. Because basketball is a free-flowing game, there exist actions in which even experts like NBA players and coaches might disagree on whether they are on-ball screens or not.

VI. Classification

Using a set of actions and their labels, a support vector machine (SVM) is used to construct a binary classifier. Beforehand, modifications are made to the SportVU data to better fit the SVM.

A. Feature Selection

After the data segmentation and annotation layers, the size of the data is manageable. However, actions contain raw Cartesian coordinates for the ball-handler, ball-defender, screener, and the ball, which are meaningless without some additional context. To give the coordinates more meaning from a basketball point of view, some characteristics of the action are calculated from the coordinates. These characteristics, called (features), are built using basic 2-dimensional physics, including Euclidian distance and velocity. In all, 41 continuous floating points are calculated for each action using its coordinate values.

1) Pairwise Interactions

As players move dynamically on the basketball court, distances and changes in distances over time between two of them can be considered pairwise interaction features; there are three players and one static object of interest: ball-handler (BH), ball-defender (BD), screener (SC), and basket (HP), resulting in a total of six combinations of two points to be considered. The

features in this section have been described first by McQueen et al. [4].

Let p_t represent the $\{x, y\}$ position of a point p at time t, where $t = 1 \ldots n$ and $n = |action|$. Let $d_{t,a,b}$ represent the pairwise Euclidian distance between points a and b at time t, where $d_{t,a,b} \geq 0$. Then, let the screen moment sm be time t at which the distance between BD and SC is at a minimum for a given action. The screen moment sm represents the exact frame in which the on-ball screen "occurs"; that is, when BD runs into the SC and the on-ball screen is producing its maximum intended effect.

$$sm = \underset{1 \leq t \leq n}{\operatorname{argmin}} d_{t,BD,SC} \tag{1}$$

sm is the point that divides an action into two slices: the approach and the departure. The approach can be defined as the short-term setup that leads to the on-ball screen, and the departure as the short-term consequence that results from the on-ball screen. Let $sm - 1$ be the duration of the approach, where $t = 1$ refers to the first frame of the action. Let $n - sm$ be the duration of the departure, where $n = |action|$ and $t = n$ refers to the last frame of the action.

Four features are selected based on sm: the average distance between two points during the approach, the average net relative speed between two points during the approach, and the same two features for the departure. One additional feature, the minimum distance between the two points during the entire action, is selected as well. As an example, the minimum distance feature for the pair $\{BD, SC\}$ is the distance between BD and SC at sm.

$$\frac{d_{sm,a,b} - d_{1,a,b}}{sm} \tag{2}$$

(2) is the average net velocity between a and b during approach.

$$\frac{1}{sm} \sum_{t=1}^{sm} d_{t,a,b} \tag{3}$$

(3) is the average distance between a and b during approach.

$$\frac{d_{n,a,b} - d_{sm,a,b}}{n - sm} \tag{4}$$

(4) is the average net velocity between a and b during departure.

$$\frac{1}{sm} \sum_{t=sm}^{n} d_{t,a,b} \tag{5}$$

(5) is the average distance between a and b during departure.

$$\min_{1 \leq t \leq n} d_{t,a,b} \tag{6}$$

(6) is the minimum distance between a and b during the action.

Since there are six combinations of pairs and five features per combination, this feature selection method yields 30 features. In general, these features are looking to highlight player interactions whose changes are triggered by the on-ball screen.

2) Duration Features

The durations of the approach, departure, and the entire action account for 3 of the 41 features.

$$sm - 1 \tag{7}$$

(7) is the duration of the approach.

$$n - sm \tag{8}$$

(8) is the duration of the departure.

$$n - 1 \tag{9}$$

(9) is the duration of the action.

These features are used to handle outliers in approach or departure durations. Because those durations are used as divisors in their calculation, skew can occur if those durations are close to zero. Using the durations as features themselves helps to mitigate that skew.

3) Setup Features

The 33 features described above do not take the location of the ball into consideration. Because the on-ball screen occurs without a pass, the distance between the ball and the ball-handler is minimal. BH is chosen over the BL simply because the coordinate data for BH encounters less errors and artifacts than BL. However, 8 additional features described below do in fact use BL as a point of interest to track passes prior to the on-ball screen.

Some of the actions labeled not on-ball screens had some distinct features not captured by the rule-based algorithms or the feature selection methods described above: the ball-handler receiving BL through a pass from SC immediately prior to the action, called a dribble handoff, which disqualifies the action from being a true on-ball screen. To account for this, an additional 13 frames prior to the start of the on-ball screen as marked by the rule-based algorithm (setup) are retroactively recorded.

Pairwise features, namely maximum and minimum distances, and average relative speed and distances, between BH and BL, and SC and BL are calculated.

$$\max_{-12 \leq t \leq 1} d_{t,a,b} \tag{10}$$

(10) is the maximum distance between a and b during setup.

$$\min_{-12 \leq t \leq 1} d_{t,a,b} \tag{11}$$

(11) is the minimum distance between a and b during setup.

$$\frac{d_{-12,a,b} - d_{1,a,b}}{n - sm} \tag{12}$$

(12) is the average net velocity between a and b during setup.

With two pairwise interactions and four features, these 8 features complete the set of 41 features for an action.

B. Discretization

All 41 features selected from an action are continuous floating points. They are discretized into bins indicated by integers for several reasons: outlier handling; reduction of data size; and data type conversion to fit the binary classifier.

For each feature, the continuous values of all labeled actions are sorted. Then, the values are subdivided into five bins using six cutoff lines, where the first and last are minimums and maximums for that feature, respectively. Each resulting bin contains the same number of actions, which are grouped based on the original continuous value for the feature. Since each bin is represented equally, this discretization method is called equal-frequency discretization.

The main advantage of equal-frequency discretization is outlier handling. If there are a few actions with features that are far off the median, the edge bins simply expand to include them, without destroying the other bins' widths.

C. One-Hot Encoding

One-hot encoding converts an integer into an array of Boolean values. Each feature, discretized into an integer between 0 and 4, is replaced by an array of size 5, one value of which is set to true depending on the corresponding integer, and the rest which are set to false. After one-hot encoding the discretized features, each action is represented by an array of 205 binary values.

D. Support Vector Machine

The labeled set of actions and their corresponding 205 features is divided training and testing sets. The training set is used to produce a 204-dimensional decision boundary by maximizing the minimum distances between each action's features and the boundary. The testing set is used to evaluate the decision boundary against the annotations. The decision boundary hyperplane is the binary classifier produced by the support vector machine (SVM).

With the given labeled actions, there is no hyperplane that perfectly splits the two groups of actions; the two classes are not linearly separable. Instead of failing, a generic loss function is applied to the SVM, where the value of the incorrect classification of an input vector is proportional to the distance from the boundary. This defines the soft-margin SVM.

VII. Results

A. Data Segmentation

In order to maximize the diversity of on-ball screens and actions, 28 games representing all 30 NBA teams were selected. For each game, only one half of the playing time were labeled.

In total, 2853 actions were identified by the rule-based algorithm. By reviewing the SportVU data and game tape, 13 on-ball screens were found to have been rejected as an action by the algorithm. That amounts to a sensitivity of 99.55%.

B. Classifier Evaluation

Results from the cross-validated linear SVM binary classifier are excellent: 90.46% sensitivity and an area under receiver operating characteristic of 0.9110.

TABLE II. Confusion Matrix for SVM Classifier

		Predicted condition	
	Total Population	prediction positive	prediction negative
	2125	**1131**	**994**
True condition	Condition positive	True positive	False negative
	1090	**986**	**104**
	Condition negative	False positive	True negative
	1035	**145**	**890**

1) Cross-Validation

When using a small set of labeled samples on which the SVM is trained, the labeled samples may not be representative of the other samples not yet seen. This phenomenon is called overfitting, to describe a classifier whose weight vector was developed using samples that were too specific, uniform, or unrepresentative. Cross validation mitigates the risk of overfitting by shuffling the labeled set into multiple different training and testing sets, then comparing results between each.

In the annotation layer, games to be labeled were chosen carefully to defeat overfitting. Games were selected to represent all 30 teams, different quarters, and dates. However, the risk still exists and must be accounted for. In general, cross validation trains and tests the SVM using some subset of the labeled samples, thereby simulating the effect of testing the classifier on unseen data. For The Framework, a cross validation method called stratified K-fold split is employed.

K-fold split is an effective method for cross-validation: split the labeled set into k number of groups, called folds, and use all but one fold to train the SVM; then test the classifier on the remaining fold, reiterating k times, until each fold was used as a test set. The scores are averaged for each iteration.

Stratification accounts for the imbalance in class distribution. In case the labeled set is grouped by class labels, stratification shuffles them into each fold evenly, making sure that each class is represented equally in each fold. Since each class is represented roughly equally in the labeled set, the benefit of stratification is minimal, but still used for a slight bump in performance.

2) Receiver Operating Characteristic (ROC)

ROC is a property of a binary classification system that is used as a diagnostic tool for its performance. It is created by plotting the sensitivity (also called True Positive Rate, or TPR) as a function of the false positive rate (FPR) by controlling for the FPR threshold. FPR refers to the proportion of actions that are not on-ball screens that are erroneously predicted to be on-ball screens.

The ROC curve is often used to get an overview of a classifier's performance under different constraints without an extensive review of the cost-benefit analysis. A perfect classifier approaches the point $\{0, 1\}$ in Figure 16. Random guess is delineated by the dotted blue line, and The Framework classifier's actual performance curve is drawn in orange. According to the ROC Curve, the classifier quickly approaches high sensitivity when the threshold for FPR is low. The dotted grey line indicates that the classifier achieves more than 83% sensitivity when the threshold for fall-out is at 10%.

This characteristic can be described using a single floating point by calculating the area under the ROC curve (AUROC). A classifier with an AUROC of 1.0 is perfect; one with AUROC of 0.5 is as bad as a random guesser. The Framework produces a classifier with an AUROC of 0.9110.

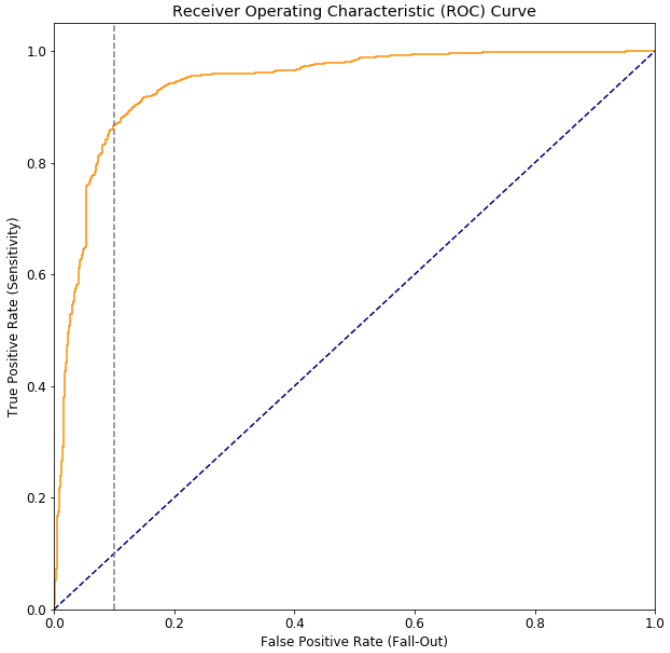

Fig. 6. Algorithm for finding the screener

VIII. Conclusion

The framework reconstructs every function of the end-to-end technology that enables in-depth basketball analytics using SportVU data while improving upon existing systems. It also provides additional features that were previously unavailable, including animated court visualizations and software extensibility.

Limitations of the framework include its reliance on individual human observations and intuitions about the game of basketball to train the classifier, which is susceptible to biases that are not always transferrable to the game as a whole. The perspectives of professionally trained basketball experts may improve the performance of the framework.

Another possibility in mitigating the limitations is to employ unsupervised learning to provide insights into patterns of basketball play that may or may not have been well understood by the traditional basketball lexicon.

IX. References

[1] Paul Dickson. The Joy of Keeping Score. New York: Walker. ISBN 0-15-600516-6.

[2] Nick Fasulo. MIT Sloan Sports Analytics Conference, Day 1: Explaining The 10,000 Hour Rule, And More. SBNation.com. Retrieved 2013-03-04.

[3] Avery McIntyre, Joel Brooks, John Guttag, and Jenna Wiens. Recognizing and Analyzing Ball Screen Defense in the NBA.

[4] Armand McQueen, Jenna Wiens, and John Guttag. Automatically Recognizing On-Ball Screens. In 2014 MIT Sloan Sports Analytics Conference, 2014.

[5] Basketball Data Feed | Basketball Player Tracking | SportVU, http://www.stats.com/sportvu-basketball-media/, Retrieved Dec. 2016.

[6] NBA.com/Stats | FAQ, http://stats.nba.com/help/faq/, Retrieved Dec. 2016.

[7] NBA partners with Stats LLC for tracking technology, http://www.nba.com/2013/news/09/05/nba-stats-llc-player-tracking-technology/, Sep 5, 2013, Retrieved Dec. 2016.

[8] Selenium Documentation, http://www.seleniumhq.org/docs/, Last updated on Apr 26, 2017, Retrieved Apr 2017.

[9] Sports Reference Terms of Use, http://www.sports-reference.com/termsofuse.html, last updated on Jan 27, 2015, Retrieved Apr 2017.

[10] Official Rules of the National Basketball Association 2013-2014, http://www.nba.com/media/dleague/1314-nba-rule-book.pdf, Retrieved Apr 2017.

[11] LIBLINEAR -- A Library for Large Linear Classification, https://www.csie.ntu.edu.tw/~cjlin/liblinear/, Retrieved Apr 2017