CLASSIFYING DEFENSIVE BASKETBALL PLAYS USING MACHINE LEARNING

A Special Project Presented to the
Faculty of the Department of Computer Science,
University of the Philippines Cebu

In Partial Fulfillment
Of the Requirements for the Degree
Bachelor of Science in Computer Science

By
Arnold Joseph Caesar P. Esteban
May 2017

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In basketball, knowing the defensive strategy of the opponent is one of the things that can help teams strategize their offense. However, defensive strategies are not recorded in the box score, or in play-by-play data. This situation forces teams and scouts to resort to time consuming activities, such as watching games and tapes. In 2013, NBA has entered the big data scene upon the installation of SportVU cameras to all NBA arenas. SportVU cameras track players at a rate of 25 frames a second, enabling detailed analyses of player movement patterns. Since then, several studies have used the SportVU data to better understand the game. However, most of the studies focused on investigating the offensive strategies of teams and building metrics for offense, and some studies that investigated defensive efficiency of players. This study presents five SVM (RBF kernel) classifiers trained with features extracted from the SportVU data in order to classify zone defense and man to man defense plays. Five experiments were performed, using individual based features and team based features. One experiment applied feature selection by using L1-Based Feature Selection with Linear SVM as the estimator. Almost all models generally perform well. The model with a combination of team and individual based features (8 team based features and 5 excess Voronoi area from defenders) was seen as the best performer among all models, with a G-mean score of 0.9293. This study can become a component towards the development of an automatic game analysis tool using the SportVU data, as this can give information about the defense plays that the NBA teams ran, without the need of watching every game.

Keywords: *Basketball defense, SportVU, SVM, Feature Selection*

TABLE OF CONTENTS

Page

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

"In all the research you do as a coach, studying other coaches and championship-type situations, you find that all those teams combined talent with great defense. You've got to stop other teams to win." – Pat Riley (Mac, 2010)

Great offense comes with great defense. Defense hustle plays, average speed, and defensive impact can change the momentum of a game. Knowing the defensive strategy of the opponent will definitely help teams strategize their offense. It will also help scouts give more comprehensive and detailed reports. Teams have to study opponent's defensive strategies in order to plan which offensive play to use. However, these do not show up in the box score - a tally of players' steals, rebounds, blocks, assists, scores which is simply a record of events known to "summarize" the entire game. Currently, scouts, teams have to watch the entire game to check on what really happened on the court. For the players and coaches, they have to watch past films of their games and the games of their opponent to assess themselves and strategize (National Basketball Association [NBA], 2016, 4:10). While it helps them see the strategies and the loopholes of the other teams, it would really take much time to analyze many things. Furthermore, Franks and Miller (1986) found during their experiment that even basketball experts can only recall as low as 42% of the significant events of a basketball match. This tedious analysis is a subject for research and automation. Several areas of the films can be examined and annotated via automation.

Automation could give a data-driven report, helping the teams know about the strategy of the other team. This can be done through computer vision and machine learning.

One classic example of activities that needs automation is surveillance. CCTV cameras can record activities in a particular area of an establishment. However, recording does not mean anything unless reviewed and observed by a person. One of the solutions that can be done is to hire a CCTV operator who will sit in front of video monitors and observe the video output for several hours a day. In an experiment by Sandia National Laboratories in 1979 (1999, p. 30, par. 2), it was found out that after 20 minutes, of watching and monitoring video output, the attention of the participants who were tasked to watch and evaluate activities degenerated. In the context of game analysis, this processing deems futile.

Machine learning techniques makes automation possible. Machine Learning is an emerging field in the new IT (Accenture, 2016, p.5). Machine learning algorithms can automate human tasks as long as the algorithms are trained well with data. Several companies are now offering intelligent video analytics platforms. These platforms automate human task to remove human attention degradation and maintain consistency in processing. IBM (2015) created IBM Intelligent Video Analytics, a software that identifies attributes, events, and patterns of behavior in real-time and converts the video images into data that can be filtered and analyzed. NEC Corporation (2012) on the other hand, created a software called NeoFace that does facial recognition designed to optimize surveillance and monitor movement and volume of people in public areas.NBA has entered the big data scene upon the installation of SportVU cameras to all NBA arenas in 2013 (NBA, 2013a). The cameras track players' {X, Y} movements in the court 25 frames per second.

Furthermore, the cameras track the movement of the ball in three dimensions (Stats LLC, 2016). This spatio temporal data provides "a continuous stream of innovative statistics based around speed, distance, player separation, and ball possession" (NBA, 2013a), making it possible to answer questions that were not answered by the traditional box score statistics ("Big Data meets big-time basketball", 2014, par. 3).

Since then, several studies have used the SportVU data for improved analysis. Maheswaran, Chang, Henehan, and Danesis (2012) analyzed the lifecycle of a rebound, discovering the optimal placement of players to obtain a rebound. Goldsberry and Weiss (2013) on the other hand used the data to characterize and understand the interior defense of the games in NBA. Maymin, Maymin, and Shen (2012) used SportVU data to examine the physics of the free throw shots of each basketball players, and found out that the causes of success and failures in free throw shooting depends on the individual. Wiens, McQueen, and Guttag used the SportVU data to create an SVM classifier that can recognize whether an on-ball screen occurred when 2 players (the ball handler, and another teammate) are closest to each other. Wang and Zemel (2016) used the SportVU data to classify offensive plays in the NBA using a variant of Recurrent Neural Networks (RNN) called Long Short Term Memory (LSTM) Networks. McIntyre, Brooks, Guttag, and Wiens (2016) used the output data, identified on ball screens from the SportVU data, of the previous work of Wiens et al (2014) as their input data in order to classify the defensive scheme of the defending team on the on ball screen. Cervone, D'Amour, Bornn, & Goldsberry (2014) created a metric called Expected Possession Value (EPV). EPV is an offensive metric at the individual level that "assigns a point value to every tactical option available to a player at each moment of a possession." Franks, Miller, Bornn, and GoldsBerry (2015) used the

SportVU data to create Counterpoints - an estimate of the points scored against a particular defender.

The studies made on top of SportVU data improved several aspects of game analysis that cannot be normally done using the previous game analysis techniques. The creation of SportVU data in fact changed the methods toward using machine learning techniques. However, the studies mainly focused on the offensive aspect of basketball: classifying certain offensive plays, creating metrics for offenders. While some studies focused on the defensive aspect of basketball, the studies did not account for the defensive play at a team level perspective. Thus, classifying the defense play of the defending team using the SportVU data remains as an area that is to be explored.

1.2 Research Objectives

1.2.1 General Objective

The general objective of this study is to be able to classify basketball defense plays from SportVU data.

1.2.2 Specific Objectives

Specifically, this study intends to:

1) Develop a methodology for processing SportVU data as inputs to the machine learning classifier,

2) Build an SVM classifier based on the inputs obtained from (1) to create defensive play classifier, and

3) Evaluate the performance classification model in classifying the defensive plays of NBA teams.

1.3 Scope and Limitations

Due to the various types of defense, this study is delimited to identifying only two types defensive play: man-to-man defense or zone defense. Only half court types of defense will be classified. Furthermore, one assumption in this study is that any play that is not man-to-man will be considered coarsely as zone defense.

This study used 266 instances of half court plays extracted from the 8 games with SportVU data in the 2015-2016 NBA Season. 48 of the instances depict the defensive team playing zone defense while the other 218 instances depict the defensive team playing man-to-man defense.

Another assumption in this study is that the defensive play of the defending team during a possession does not change. This means that the defenders will not switch from man-to-man to zone, vice versa, and switching from a zone to another type of zone defense during the course of the possession. The possessions selected from this study is only delimited to plays that started from an inbound play after timeouts, and violation calls such as out of bounds, fouls, travelling, 8 second violation, 3 second violation on offensive team. This is to ensure that the defensive team has established their positioning on the court

1.4 Significance of the Research

This study contributes to the existing researches on game analysis, more specifically analysis of SportVU data. SportVU cameras tracks players' {x,y} coordinates on the court, the ball's {x,y,z} location at a rate of 25 frames per second. These tracked coordinates and location, and timestamp comprises the SportVU data for one game. Although this data unlocks better analysis of the basketball games, the rawness of the data poses a challenge

to analysts to create data driven insights. In the field of data mining and machine learning, SportVU data incorporates instantaneous tracking, that is also embedded on a spatio-temporal dimension poses a big challenge on using many automation techniques in pre-processing so that it could be fed into Machine Learning algorithms in order to build models.

This study will contribute to existing researches in terms of extracting meaningful features from the SportVU data in order to be fed into a machine learning model in order to classify defensive plays. Several studies have used the SportVU data to classify offensive plays, create real-time metrics, reaction of defense in a pick and roll, and rebounds, but no study has contributed to understanding defensive tactics of teams by classifying their defensive plays. The model and subsequent classification provides a foundation for further analysis of games using the SportVU data.

Furthermore, this study can become a component towards the development of an automatic game analysis tool using the SportVU data, as this can give information about the defense plays that the basketball teams ran, without the need of watching every game. For example, we would know which teams run man-to-man defense often, or which teams run zone defense often, and what defense play do they execute over the course of a game.

CHAPTER 2

REVIEW OF RELATED LITERATURE

This chapter is divided into the four parts: Basketball: Game and Data, Player Tracking data (SportVU data) and Related Studies, Support Vector Machine, and Feature Selection

2.1 Basketball: Game and Data

2.1.1 Basketball

According to the International Basketball Federation (FIBA, 2014), a basketball game is "played by 2 teams of 5 players each. The aim of each team is to score in the opponent's basket and to prevent the other team from scoring." This means that one team does offense while their opponent does defense. The winning team is determined as the team that got more points after the time expires.

Article 2 of FIBA's rule book states the parts and dimensions of the basketball area where each team's basket and area are located called court. The playing area of the court is delimited by boundary lines. At the half of the court is the center line which marks the back and the front courts. Each team owns a back court. The court where the basket that the offenders attack is located is called the front court. During a normal play, all players are on one half of the court, where the offense tries to score while the opponents prevent them from scoring. This is due to the eight second rule which states that the offensive team "shall not be in continuous possession of a ball which is in its backcourt for more than 8

consecutive seconds" (NBA, 2013b, p. 36), and the rule wherein the offensive team cannot bring the ball back to their backcourt once they have brought the ball to the frontcourt.

2.1.2 Defense

Defense is the action of preventing the offense from scoring against their basket. This means that no player in the defending team possess the ball. Each defender is assigned to guard at least one offensive player, following wherever the offender goes. This type of defense is called the man-to-man or the man-marking defense. An alternative to man-to-man defense is the zone defense, where each players are assigned to guard specific zones or areas of the court rather than offensive players (Gudmundsson, & Horton, 2017). A combination of these types of defense are called hybrid zones or junk defense, where some players do man-to-man while others do zone defense (Parker, 2015). One example of junk defense is the box-and-one (four players form a box like zone while one player follows his man throughout). Figure 2.1 illustrates a Venn diagram showing the relationship between the 3 main types of defensive plays.

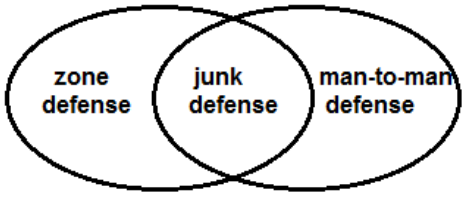


Figure 2.1. Venn diagram of the defensive plays. Junk defense is a combination of man-to-man and zone defense.

The articles by Parker (2015) and Zillgitt (2012) discuss as to how and why NBA teams frequently use man-to-man defense or junk defense than pure zone defense. For years, NBA did not allow zone defense. Players, when caught by referees not actively guarding

any player, gets called for an illegal defense violation (Parker, 2015). NBA has now allowed zone defense, but the defensive three second violation (NBA, 2013b, p. 36) makes it difficult for teams to execute zone defense. This means that NBA teams rarely play pure zone defense in the NBA.

2.1.3 Box Score

A box score is a summary of an NBA game, displaying the major statistical categories for every player of both teams (NBA, 2001). Major statistical categories denoted with abbreviations are defined as follows: (a) MIN means minutes played; (b) FGM means field goals made in the 2-point and 3-point area; (c) FGA means field goals attempted in the 2-point and 3-point area; (d) FTM means free throws made; (e) FTA means free throws attempted; (f) OFF means offensive rebounds; (g) DEF means defensive rebounds; (h) TOT means the sum of offensive and defensive rebounds; (i) AST means assists; (j) PF means personal fouls; (k) ST means steals; (l) TO means turnovers; (m) PTS means the total points of a player from 2-point field goals, 3-point field goals, and free throws; and (n) TOTALS means team totals.

While it is true that the major statistical categories show the summary of the game, however, the statistics are only recorded after a possession ends. It cannot account for the defensive strategy, positioning, and spacing of the players. A sample of a box score data can be seen on Appendix A.

2.1.4 Play-by-Play data

Play-by-play statistics are an expansion of the box score statistics (Puranmalka, 2013). It contains the time information about the individual box score statistics. Furthermore, play-by-play statistics contain the following information: (a) times when player substitutions are made, (b) counter-party in the individual box score statistics, (c) type of field goal attempted (jumper, dunk, or lay-up), (d) times during possession changes. Although the time when shots, steals, and other box score statistics happened, it still could not account for the defensive strategy, positioning, and spacing of the players since the location of the players are not recorded in this statistic. Thus, this data could not give information as to which defensive play the teams executed throughout the game. A sample of a Play-by-Play data can be seen on Appendix B.

2.2 Player Tracking data (SportVU data) and Related Studies

Basketball is a spatial and temporal sport. One of the limitations of the traditional and play-by-play data is that it does not capture the entire moment that happened on the court in terms of space. Furthermore, the data recorded and quantified are the things that happened at the end of a possession, such as steals, rebounds, turnovers, fouls, blocks, assists, and shot attempts.

SportVU cameras track players {x,y} locations 25 frames a second, and the ball's {x,y,z} location (Stats LLC, 2016). Furthermore, SportVU data is annotated with play-by-play data (Maheswaran et al, 2012). This means that every play in the play-by-play data corresponds to a set of movements from the SportVU data. These sets of movements that

correspond to the play-by-play data are called moments. Hence, the player tracking data generated by SportVU is able to record the entire game's player and ball location.

Several studies have used the player tracking data to detect basketball plays. Wang and Zemel (2016) used the SportVU data to classify offensive plays in the NBA using a variant of Recurrent Neural Networks (RNN) called Long Short Term Memory (LSTM) Networks. The task of classifying NBA plays is the same with action recognition. Using the SportVU data, they were able to generate a pictorial representation of an action by mapping out the {X, Y} coordinates of the offensive team for a certain periods of time. The generated images were then used as inputs for the LSTM.

Wiens et al (2014) created an SVM classifier that can recognize whether an on-ball screen occurred when 2 players (the ball handler, and another teammate) are closest to each other: this is known as the "screen moment". They segmented the SportVU data into short periods of time called actions. Segmenting the data into actions was done by examining every frame against a set of criteria described in their rule-based algorithm. Average action time is about 1.5 seconds. For each action, the pairwise distances between the ball-handler, screener, and defender were obtained. To extract features, the action was split into two halves: the approach and the departure. The approach happens when the ball-handler and the screener are moving towards each other while departure happens after - when the ball-handler is moving past the screener. The other features represent the velocity of the players involved in a screen in relation to each other. Data point was labeled as +1 if a ball screen happened while -1 if there was no ball screen. They built a Support Vector Machine (SVM) classifier to determine whether a screen happened or not in that action.

McIntyre et al (2016) used the output data, identified on ball screens from the SportVU data, of the previous work by Wiens et al (2014) as their input data in order to classify the defensive scheme of the defending team on the on ball screen. For each example, features were extracted based on the pairwise distances between the ball handler, on ball defender, ball screener, and the screener defender. From the set of pairwise distances, called time series signal, summary statistics were selected as features. However, the time series signal varied in length for each example. Some examples had longer time series signal length than other examples. In order to obtain a fixed time length for all examples, the examples were divided into ten equal length segments, and features were extracted from each segment to incorporate temporal information. Features extracted were summary statistics describing the signal (e.g., Histogram of each segment, slope, and the quantiles of each segment). These features were concatenated, resulting into a fixed length feature vector for each example. While this takes into account the defensive strategy of the defending team, it only takes into account the players who got involved in defending the on ball screen: namely the defender of the ball handler, and the defender of the screener. This does not take into account the defensive play at a team level perspective.

Kempe, Grunz, & Memmert (2014), on the other hand, tracked players movements using the Ubisense Tracking System and obtained a spatio temporal data same as SportVU. The spatio temporal data was used to classify preselected basketball offensive plays (fastbreak, horns, and high pick) using a type of Neural Network called Self Organizing Maps (SOM). The approach by Kempe et al (2014) in preprocessing the data was different with the method used by Wiens et al. (2014). In order to remove artefacts, several {x,y} coordinates of players were averaged to one {x,y} coordinate every second, and in order

to synchronize coordinate data, the movements of the other team when playing their offensive play were mirrored, making the left and right movements of both teams when doing their offensive play congruent.

Other studies created novel metrics based on the SportVU data. Cervone et al (2014) created a metric called Expected Possession Value (EPV). EPV is an offensive metric at the individual level. EPV "assigns a point value to every tactical option available to a player at each moment of a possession." This allows analysts to evaluate each decision a player makes. For example, when a player passes the ball to an open man for three, his EPV increases, but when he shoots with two defenders in front of him, his EPV decreases.

Franks et al (2015b) used the SportVU data to create Counterpoints - an estimate of the points scored against a particular defender. They built a Hidden Markov Model using the SportVU data to express the evolution of defensive matchups over time. This allowed them to know who is responsible in defending the shooter at any given time. They then took into account the shooting ability of the shooters. They used their model to define their metric. Although this metric accounts for how well an individual defends shooters, counterpoints metric penalizes the defender on the assumption via their defensive matchup. This metric assumes that the defense played man-to-man (Franks et al, 2015a). Thus, this metric did not account for the defensive play ran by the defending team which could have affected their matchup model in determining who really is responsible in defending the shooter (Franks et al, 2015b, p 7, par 4). Using their model, many types of analyses can be conducted. One example is defining defensive entropy. Defensive entropy is defined as "the uncertainty associated with whom a defender is guarding throughout a possession" (Franks et al, 2015a, p 9). This reflects how active a defender is on court in terms of double

teams and switches. Furthermore, their model is theoretically simple to include latent variables and model defender position as a mixture model over possible defensive play in order to infer whether the defenders played man-to-man defense or zone defense (Franks et al, 2015a, p 10, par 1). Table 2.1 shows a summary of studies related to using SportVU data.

Table 2.1. Problems solved by previous studies that have used the SportVU data

Authors	Problem(s) Solved
Wang and Zemel (2016)	Classifying offensive plays using LSTM
Wiens et al (2014)	Recognizing on-ball screens
McIntyre et al (2016)	Identify type of on-ball screen defense
Cervone et al (2014)	EPV – offensive metric that evaluates player's decision at any given point in time
Franks et al (2015b)	Counterpoints – estimate of the points scored against a particular defender to characterize man-to-man defensive plays
Maheswaran et al (2012)	Characterization of rebound play of games in NBA
Goldberry and Weiss (2013)	Charactization of interior defense of the games in NBA
Maymin et al (2012)	Examine the physics of free throw shots of each basketball players

2.3 Support Vector Machine

Support Vector Machine (SVM) classifier, also known as maximum margin classifier, is a state-of-the art supervised machine learning model that has the trait of avoiding over-fitting (Bishop, 2006). The goal of SVM classifier is to find the optimal linear separating hyperplane of a binary labeled dataset. Figure 2.3 illustrates this separating hyperplane. This separating hyperplane is the one with maximum distance from the nearest training patterns (Duda, 2006). If data is not linearly separable, the kernel method can be applied to find a nonlinear boundary. Kernels accept two feature vectors from two data points and returns a similarity score between the two data points. This means that the features will be replaced by a similarity measures derived from a kernel function. Some of the commonly used kernels are Linear kernels, Gaussian kernels and Polynomial kernels. Linear kernels do not change anything on the original feature vector. The equation for Gaussian kernel is described in Figure 2.2.

$$\exp(-\gamma|x - x'|^2)$$

Figure 2.2. RBF Kernel Formula

In order to get the best results, the gamma γshould be parameterized to get the best results. Complicated kernels such as Gaussian and Polynomial kernels are used when the number of features is less than the number of training examples, and in the absence of the designer's knowledge of the problem domain. SVMs are deterministic in the sense that they just assign new examples to one class or another. Furthermore, SVM does not attempt to model the underlying probability distributions.

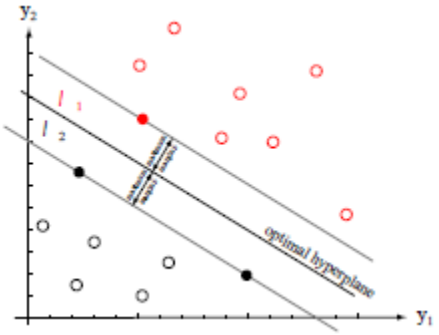


Figure 2.3. SVM's hyperplane. SVM finds the optimal linear separating hyperplane of a binary labeled dataset. The three support vectors are the solid dots (Duda, 2009)

2.4 Feature Selection

Irrelevant features, or features that do not really separate one class from the other may unknowingly be used and therefore confuses machine learning systems (Witten, Frank, Hall, & Pal, 2017). This results to deteriorating performance for the classifiers. Hence, feature selection is introduced. Aside from filtering out irrelevant features and improving performance, feature selection methods helps reduce complexity of the model and provides faster classifiers. Several feature selection methods are available. Some of the feature selection techniques include filtering and embedding. Embedded methods make use of a machine learning model to determine which features best contribute to the performance of a model. Filter methods can be considered as a preprocessing step since they apply statistical measure to make an independent assessment based on the general characteristics of the data. The feature set is filtered to produce the set with the most relevant features before learning commences. When having suspicions about the interdependence of the features, filtering for subset selection can be done. However, the problem with some filters is that the features selected might not be tuned for a machine learning model. Hence, using a wrapper or embedded method with a linear model could act as a filter seems reasonable.

The feature subset selected from the embedded method can be used to train a complex non-linear model (Guyon, & Elisseeff, 2003). An example of this approach is found on the paper of Bi, Bennett, Embrechts, Breneman, & Song, (2003), where they used a linear l-norm SVM for feature selection but a nonlinear l-norm SVM was built for training and creating the classifier.

CHAPTER 3

METHODOLOGY

This chapter is divided based on the pipeline of work, namely: data, data segmentation, preprocessing and feature selection, data labelling and splitting of data, building the defensive play classifier (SVM), and evaluation of the defensive play classifier. The output for this work is a defensive play classifier. The pipeline of work for this study is shown in Figure 3.1.

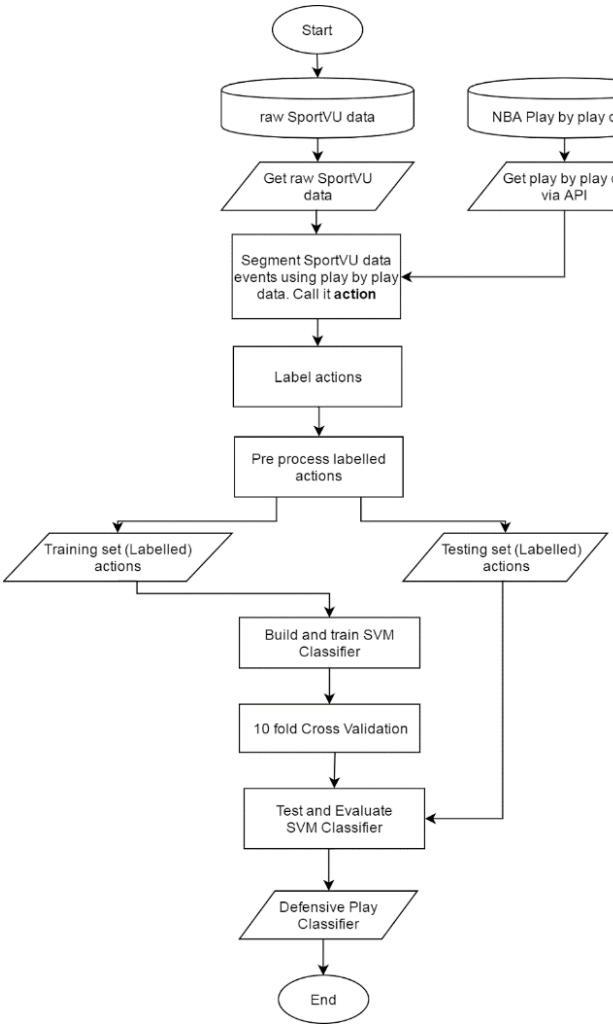


Figure 3.1. Flow of work. This serves as the framework that has to be followed in order to build a defensive play classifier.

3.1 Data: SportVU data and play-by-play data

The first step is to obtain the SportVU data from from GitHub repository of Johnson (2016). This GitHub repository contains the first 637 games of the 2015-2016 NBA Season. For every game, there is a corresponding SportVU data in JSON (JavaScript Object Notation) format. The structure of the SportVU data in JSON format is described in Appendix C.

By default, the SportVU data is mapped according to play-by-play data. Every moment in the SportVU data corresponds to a record in the row set of play-by-play data. Thus, obtaining the play by data is necessary to easily segment the SportVU data. A snippet of a play-by-play data of a game is shown in Figure 3.2. The play-by-play data of a game can be retrieved via API (Application Program Interface). The API Endpoint is defined in Table 3.1. The data retrieved via API is in JSON format. The values of some important keys of the play-by-play data is shown in Appendix C. The structure of the play-by-play data in JSON format is described in Appendix D.

Table 3.1. Definition of API for play-by-play data retrieval

Endpoint:	http://stats.nba.com/stats/playbyplayv2	
Method:	HTTP GET	
Parameters:	EndPeriod EndRange GameID RangeTypes Season SeasonType StartPeriod StartRange	0 0 gameid (ex: 0021500391) 0 2015-16 Season 0 0
Sample Usage (GET play-by-play data for all quarters with Game ID 0021500391)	http://stats.nba.com/stats/playbyplayv2?EndPeriod=0&EndRange=0&GameID=0021500391&RangeType=0&Season=2015-16&SeasonType=Season&StartPeriod=0&StartRange=0	

Detroit Pistons		Chicago Bulls
Start of Q1		
	12:00	Jump Ball Gasol vs. Drummond: Tip to Rose
Caldwell-Pope STEAL (1 STL)	11:46	Rose Bad Pass Turnover (P1.T1)
	11:29	Snell P.FOUL (P1.T1) (J.Capers)
MISS Marc Morris 19' Jump Shot	11:19	
	11:17	Butler REBOUND (Off:0 Def:1)
	11:11	MISS Rose 1' Running Layup
Drummond REBOUND (Off:0 Def:1)	11:08	

Figure 3.2. A snippet of play-by-play data with Game ID 0021500391. Retrieved from NBA Stats website (National Basketball Association, 2015)

3.2 Data segmentation

In order to create a set of discrete data points for training and testing the defensive play classifier, the data has to be segmented based on this study's limitation: half-court type defense. Segmenting the data reduces the size of the SportVU data and eliminates irrelevant data. Adapting the term used by Wiens (2014), the segmented data for this study which will undergo preprocessing will be called actions. In other literature action is known as possession.

Several studies have their own implementation of segmenting the SportVU data that suits the criteria met for their machine learning model. By default, SportVU data is composed of set of movements called "moments" that are mapped according to play-by-play data. Wang and Zemel (2016) discarded in-bound, after timeout plays, and transition offense plays since what they were after for were normal half-court plays. In-bound and after timeout plays have distinct markers in the SportVU data. Hence, removing these plays from the candidate data set can easily be achieved by looking up from the play-by-play data. Furthermore, they claim that transition offense "is fairly easy to identify, and its nature differs from half-court strategies quite drastically" (Wang and Zemel, 2016, p. 6). The labeled data set was provided by their research grantor, the Toronto Raptors.

Wiens et al (2014) segmented data by creating a rule based algorithm to find the candidate regions that are possibly being an on-ball screen. The algorithm looks for 13 or more consecutive frames (at least 0.5 seconds long) that meet a certain set of criteria for possibly being an on-ball screen. These set of consecutive frames that meet the set of criteria are called actions. These actions were then preprocessed and used as input data for training and testing the on-ball screen classifier.

The events that Wang and Zemel (2016) segmented are quite similar with the events for the defense play classifier because offensive plays are called when all players are on one side of the halfcourt. However, in their paper, there was no mention of how the normal half-court plays were extracted. Hence, for this study, there is a need to create a rule-based algorithm that can extract a half court play by examining each frame of the SportVu data.

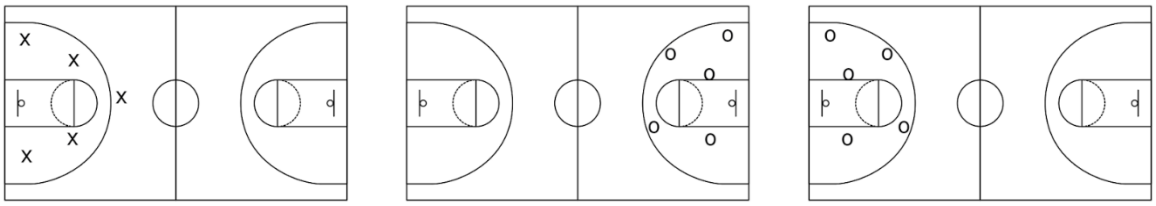
Since the positioning of the defensive team is already established on an after timeout or an inbound play by looking up at the play-by-play data mapped to the game's SportVU data, the moments of these after timeout and in-bound plays will be considered as actions. However, some frames will be to remove artefacts. The rule based algorithm is shown in Figure 3.3. First, after inbound and after timeout plays are extracted by looking at the description of the event found in the play-by-play data. The event ID of these plays will be stored to `play_list`. In the rule based algorithm, the reason why the frames where the ball is already in the paint for a long time are not considered because the type of defensive play is indistinguishable at these moments. After segmenting the data into actions, selecting features from the actions will be determined. See Appendix E to view full code.

```
extract after inbound and after timeout plays add to play_list
 for each eventId in play_list:
   initialize empty list as frame_list
   inside_count = 0
   for each frame in the play:
     [
     if all players are on one side of the court & ball is not held
in the paint & inside_count <= 10:
       add the frame to frame_list
       inside_count = 0
     if ball is held in paint and inside_count <= 10
       add the frame to frame_list
       inside_count = inside_count + 1
     if inside_count is > 10 or offense attempts to shoot
       restart frame_list
     ] for length of frame_list >= 75 & frames added were contiguous:
         identify frame_list as action
```

Figure 3.3. Rule based algorithm to segment the data into actions.

3.3 Preprocessing and Feature Selection

In order ensure that all actions occur in the same side of the court (halfcourt), the {x,y} coordinates of the other team when playing their defensive play will be transformed across the half court line without loss of generality. This makes the left and right movements (with respect to the half court line) of both teams when doing their defensive play congruent. A visualization on the effect of mirroring the {x,y} coordinates of players across the halfcourt line is shown in figure 3.4.



(a) (b) (c)

Figure 3.4. The effect of transforming the coordinates of players across the halfcourt line. Figure 3.4 (a) depicts the original location of players of one team when doing defense. Figure 3.4 (b) depicts the original location of players of the other team when doing their defense. Figure 3.4 (c) depicts the effect of transforming the other team's coordinates across the halfcourt line, ensuring that all actions occur in the same side of the court

Several features from the actions were selected. These features will then serve as inputs in training and testing the defensive play classifier. Selecting features is based on what a classifier needs to learn. In the case of recognizing on-ball screens by Wiens et al (2014), five continuous features were extracted based on pairwise distances of players involved in a screenplay from the action example: the ball handler, the on ball defender, the screener, and the screener defender. The continuous features extracted were: the screen moment- when the pairwise distance between the ball-handler and screener is at minimum; departure - when the ball-handler is moving past the screener; and the 3 other features represent the velocity of the key players in relation to each other.

In the case of classifying offensive plays by Wang and Zemel (2016), directly using the {x,y} coordinates as features are not sensible representations of an offensive play. A better input for the offensive play classifier would be an image. The image consists of points representing the locations of the players and the ball in every frame. This creates a footprint of the movement of the entire offense play. Furthermore, this simplifies the problem into an image classification problem. However, using this method, the pattern can be seen but the time was not regarded. Certain movements of players when plotted might

match with the footprint even though the play was not called. Hence, for offensive play classification, it is better to represent the input as a sequence of images where each image in the sequence is the current state of the player location. Images from the previous states are also plotted to the current image with a lower alpha level to create a shadow or trail suggesting where the players came from.

In the case of classifying defensive plays, there are important aspects of a possession needs to be looked upon, namely, the position of the defenders over time, location of the ball, and their matchups. Selecting features relevant to the nature of the defensive play have to be extracted in order to be able to differentiate between the defensive plays to be classified. The different features extracted from the SportVU data are summarized in Table 3.2.

Table 3.2. Features extracted from the SportVU data. This will serve as an input vector for the defensive play classifier.

Feature	Alias	Number of features	Category
Defensive entropy of the team	entp	5 continuous features	Man marking
Average time defenders defending their respective matchups	time	5 continuous features	Man marking
Average velocity of the offensive team	vel_off	5 continuous features	Physics
Average velocity of the defensive team	vel_def	5 continuous features	Physics
Excess area of the Voronoi tessellations of each defenders	vor	5 continuous features	Positioning and Spacing
distance from initial position with respect to the "average" position of the defenders	init	5 continuous features	Positioning and Spacing
Average distance between players and their respective canonical position	canon	5 continuous features	Distance based
Average distance between players and their respective matchups	matchup	5 continuous features	Distance Based
Difference between "matchup" and "canon"	diff	5 continuous features	Distance Based
TOTAL NUMBER OF FEATURES		45 features	

Man-to-man defense is played when all defenders never left their matchup (man) and followed wherever their man went over the course of the action. In this case, the feature extraction made by Wiens et al (2016) could be adapted. Features can be extracted from the pairwise distances between the players, and their distance with the hoop. However, this poses a problem in the sense that by looking at the pairwise distances, defensive

assignments seem to be all about proximity. This makes some plays look like “zone” even though it is a man to man defense. For example, at time t, defender 1 is defending offender 1 but at time t + 1 defender 1 defends offender 2 because “they have smaller pairwise distance compared to offender 1”. But at time t+2 defender 1’s assignment is offender 1. By looking at the interaction of the players over time, defender 1 was actually chasing offender 1, but by looking at the pairwise distance it can be inferred that the defender switched to another offender as matchup. This usually happens during an on ball screen. The on ball defender might have received a screen, leading him to have a smallest pairwise distance with the screener, and his defensive assignment at that point in time looks like it is between him and the screener although in the next time steps he is actually chasing the ball handler - his initial man. Hence, there is a need to look at the previous time stamps in order to maintain correct matchups over time.

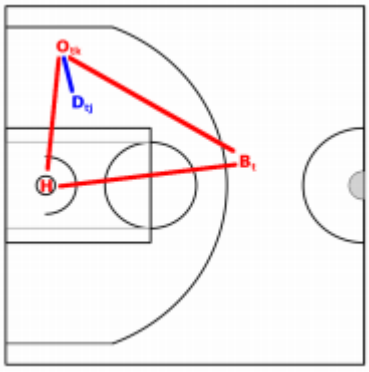


Figure 3.5. Mean location of the defender. Image from Franks et al (2015b)

Franks et al (2015b) built a model that could determine which defender is marking each attacker. For a given offensive player at a given time, the mean location of the defender was modelled as a convex combination between the location of the offender, hoop, and the

ball. Figure 3.5 shows the mean location of the defender. The location of the defender given the observed location of the offender, was modelled as a Gaussian distribution about a mean location. They built a Hidden Markov Model (HMM) in order to ensure that the matchups smoothed for each time step. This enabled them to determine switches of defensive assignments and double teams. They found out that a defender's mean location can be described as:

$$0.62O_{tk} + 0.11B_t + 0.27H$$

Figure 3.6. Defender's mean location formula.

where *Otk* is the current location of the ball at time t, *Bt* is the current location of the ball at time t, and *H* is the location of the hoop. Using this model they were able to measure the defensive entropy of the defensive team, which is defined as "*the uncertainty associated with whom a defender is guarding throughout a possession*." In terms of switches and double teams, entropy reflects how active defenders are on the court. This could be a useful feature to be included. A defender's entropy is defined as:

$$\sum_{k=1}^{5} -Z_n(j,k)\log(Z_n(j,k))$$

Figure 3.7. Entropy of the players

where $Z_n(j, k)$ is the fraction of time defender j spends guarding offender k in possession n. Defensive entropy is zero if the defender guards only a single player throughout the course of a possession. If the defender splits his time equally between two offenders, his entropy is one.

Isolation plays are known to be defended with a man-to-man defense. Defenders follow their matchups while the ball handler does a one-on-one game with his matchup. From the current perspective this play may look like a zone defense because defenders tend to be in their "zone" as their matchups tend to be stationary. However, it can be observed that the ball handler had the ball for a very long time, and might have the tendency not to pass it. The average velocity of the players could be used as a feature. This feature could help differentiate zone from man. Zone defenses are known to slow down the offensive play, hence offenders tend to have slower movements.

Positioning and spacing of the players depending on where the ball is located is an important feature to be extracted. Usually on a zone defense when the ball is passed to the wing side of the court players from the opposite wing "sag" or move near the hoop in order to help in case when the offensive team is able to bring the ball in the paint. However, sagging could also happen in a man-to-man defensive play. The difference might not lie on the positioning in this case, but rather where the front part of their body face. In a man-to-man defensive play defender's body would still face with their man, their front bodies might be slightly tilted. In the case of man-to-man, their front bodies might be more tilted towards where the ball. However, the angle to where the players face is primarily absent with the SportVU data, since the data is only composed of coordinates that could be plotted on a plane. The absence of this feature makes the classification of defensive plays very difficult. Furthermore, the rare occurrence of true zone defenses in the NBA makes it even more difficult to build a robust classifier (Franks et al, 2015a). The angle at which the defender faces could have been an important feature for the classifier. To account for positioning and spacing, initial "zone" assignments of the players can be checked. This will

be done using Voronoi tessellation. The tessellations that were built sets the zone assignments each defender has over the course of the action. The hypothesis in the study by Kim (2004) will be used. For each frame, the Voronoi area for each player will be recorded. Each player's area will be averaged, and subtracted to the initial Voronoi area (at the first frame). This is called the excess area.

3.4 Data Labelling and Splitting of Data

The actions are labelled only under two categories: man-to-man defense, or zone defense. Actions that are likely to be a pure man-to-man defense will be labelled as "-1" while "1" will be labelled for zone defense.

Labelling the data set is done manually, but it depends on who labelled the data. Wang and Zemel (2016) were provided with labelled data by their research grantor, the Toronto Raptors. On the other hand, Wiens et al (2014) manually labelled their data. Because of this, their ability to evaluate their method was limited.

For this study, data labelling was done manually. By merely looking at the JSON data, it is impossible to determine the defensive play ran by the defending team since the data itself lacks context. One way to solve this problem is to visualize the points into a two dimensional plane, where the plane represents the court while the points represent the {x,y} coordinates of the players and the ball. However, the problem of visualizing the action into a 2D plane is that the points lack context. SportVU data simply represents a player into geometric primitive, which obscures the nature of the player (Goldsberry & Weiss, 2013). Key information such as the 3D structure of the players, their defensive stance, where the players look, cannot be seen since the players are only represented as points. In order to

fully get the key information, the video clips of the actions were extracted and observed, and then the label of the action was determined.

3.5 Building the defensive play classifier: SVM

SVM is used as the defensive play classifier. SVM was chosen to be used in this study instead of extending the HMM by Franks et al (2015b) in order to build a mixture model primarily because of the following reasons: (1) this study assumes that there is no changing of defensive plays over the course of the action; (2) true zone defense is rare in the NBA (Franks et al, 2015b), making it difficult to separate zone defenses from pure man-to-man defense; (3) because of (2), the researcher chose to use a supervised learning algorithm instead of an unsupervised learning algorithm such as EM algorithm that was used in learning the HMM parameters; (4) because of the nature of labelling the data, a deterministic model such as SVM is enough to use over a probabilistic model like the Gaussian mixture model, which is a type of a mixture model where a defensive play in an action overlaps under two classifications or clusters. Hence, in this study, the model built by Franks et al (2015a, 2015b) was not extended, but some of the characteristics derived from the model were used as features for the SVM classifier.

Since the number of features (45 features) in this study is relatively smaller than the number of training examples (224 examples), a Gaussian kernel (radial basis function kernel) is used for the SVM classifier. The model is built using Scikit-learn library's SVC class which is based on LIBSVM library (Pedregosa et al., 2011). Since a Gaussian kernel is used, feature scaling has to be done. Furthermore, the class weights of the SVM is set to

"balanced" mode to automatically adjust weights based on classes. Regularization parameter C is set to 1.0.

Five experiments were set up. The goal of the experiment is to check whether reducing the number of features could still result to a good classifier, and determine which set of features would provide the best performing classifier. The first experiment checks on using all extracted features from the actions. The second experiment checks whether removing the excess Voronoi area feature would still result to a good classifier. The third experiment tries to combine the individual based features into team based features, and combine it with the excess Voronoi area feature, while the fourth experiment checks whether using only team based features would result to a good classifier.

In the aim of reducing the number of features using wrapper and filtering method, a pipeline is implemented. For the fifth experiment, a pipeline was implemented. Team based features were used. Based on the suggestion by Guyon & Elisseeff (2003), Feature selection was applied by using L1-Based Feature Selection with Linear SVM as the estimator. The features selected from the estimator were used to build an SVM with Gaussian kernel.

3.6 Evaluation of the defensive play classifier

The defensive play classifier is a type of binary classification since it classifies only 2 non-overlapping classes: man-to-man and zone. Several performance measures can be used to evaluate the defensive play classifier. The confusion matrix in Table 3.3 serves as a lookup table in evaluating the classifier and is used to properly present the classified data. Furthermore, possible trends in misclassification can be observed using this matrix. From

the confusion matrix, true positive (proportion of correctly classified examples), true negative (proportion of correctly detected examples not belonging to the class, false positive (proportion of incorrectly classified examples to a class), and false negative (proportion of incorrectly not detected examples belonging to a class) can be used to compute for accuracy, sensitivity, and specificity. These performance measures can be obtained to evaluate the defensive play classifier.

Table 3.3. Confusion matrix of a binary classification

	Classified as zone defense	Classified as man-to-man defense		
Zone	True positive	False negative	Sensitivity (Recall)	G-mean
Man-to-man	False positive	True negative	Specificity	
				MCC

Sensitivity (also known as recall) is the proportion of true positives from cases that are predicted as positive. Specificity, on the other hand, is the proportion of true negatives from cases that are predicted as negative. Since the data is composed of imbalanced classes, sensitivity and specificity measures can be used for this purpose (Han, Kamber, and Pei, 2012). Sensitivity and Specificity can be combined to calculate the Geometric mean (G-mean). G-mean is a metric that estimates the performance of the classifier on both classes. A low G-mean means a poor performance in classifying positives even if the negatives are correctly classified by the model. This metric is important in avoiding underfitting the positive class and overfitting the negative class (Hoang, Bouzerdoum, & Lam, 2009).

These three metrics are used when the performance of both classes is concerned and expected to be high simultaneously.

Matthews Correlation Coefficient (MCC) is a single performance measure based on all values from the confusion matrix and is generally regarded as a balanced measure since it considers mutual accuracies and error rates on both classes. This can be used even if the classes are of different sizes. MCC is a correlation coefficient between the predicted and observed binary classifications. A coefficient of +1 indicates perfect prediction, while -1 indicates total disagreement between prediction and observation. When MCC is close to 0 this means that the model performs randomly (Bekkar, Djemaa, & Alitouche, 2013).

The formula for these four performance measures are shown in Table 3.4. These performance measures will be computed to give an impression of the effectiveness of the SVM classifier in classifying defensive plays.

Table 3.4. Performance measures to be used in evaluating the defensive play classifier. TP denotes number of true positives, TN denotes number of true negatives, FP denote number of false positives, and FN denotes false negatives

Performance Measure	Formula	Remarks
Sensitivity	$\frac{TP}{TP + FN}$	true positive rate. higher score means the classifier is able to classify zone defense correctly 0 sensitivity 1
Specificity	$\frac{TN}{TN + FP}$	true negative rate. higher score means the classifier is able to classify man-to-man defense correctly 0 specificity 1
G-mean	$G = \sqrt{Sensitivity\ X\ Specificity}$	Low G-mean indicates poor performance in classifying positives even if the negatives are correctly classified by the model. $0 \leq$ G-mean ≤ 1
MCC	$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$	1 perfect prediction 0 random prediction -1 total disagreement on the observation and prediction $-1 \leq$ MCC $\leq +1$

CHAPTER 4

RESULTS AND DISCUSSION

In this study, five experiments based on features were done. An SVM classifier was built for each experiment. Each experiment's model was evaluated. Table 4.1 shows the features used for each experiment.

4.1 Experiments and Results

266 actions were obtained from 8 games with SportVU data and were labelled as man-to-man or zone. 48 of these actions were labelled as zone defense while the remaining 218 actions were labelled as man-to-man defense. Because of the class imbalance problem, 10 fold stratified cross validation is used with 70% of examples are under the training set while the remaining 30% were placed under the test set.

Table 4.1. Features used for the five different experiments.
See Table 3.2 to see full description. Displayed are the aliases of the features

Exp No.	Description	Features [Number of features]	Category (Individual or Team Based)
1	All individual features 45 Features	entp [5] time [5] vel_off [5] vel_def [5] avg_def [5] vor [5] init [5] canon [5] matchup [5]	Individual Individual Individual Individual Individual Individual Individual Individual Individual

2	All individual features except excess area of Voronoi tessellation 40 Features	entp [5] time [5] vel_off [5] vel_def [5] diff [5] init [5] canon [5] matchup [5]	Individual Individual Individual Individual Individual Individual Individual Individual
3	All individual features averaged + individual excess area of Voronoi tessellation (Team based features) 13 Features	entp [1] time [1] vel_off [1] vel_def [1] diff [1] vor [5] init [1] canon [1] matchup [1]	Team Team Team Team Team Individual Team Team Team
4	All individual features averaged, except for individual excess area of Voronoi tessellation 8 Features	entp [1] time [1] vel_off [1] vel_def [1] diff [1] init [1] canon [1] matchup [1]	Team Team Team Team Team Team Team Team
5	Pipeline: L1-Based Feature Selection using Linear SVM Selected features used to build SVM with Gaussian Kernel	entp [1] vel_off [1] vel_def [1] diff [1] init [1] canon [1] matchup [1]	Team Team Team Team Team Team Team

The first experiment setup was to use all features as seen on Table 3.2. The second experiment setup was to use all features used in the first experiment except for the excess area of Voronoi Tessellation feature. The third experiment setup was to represent each features (with five columns each) into one. The individual features were averaged and was

used as feature. These features will be called as “team based features”. This representation is not applicable for the excess area of Voronoi Tessellation feature since the sum of the area of each player would just result to the area of the half court. Hence the average would just be the same for all examples. The fourth experiment uses all of the features from the third experiment except for the excess area of Voronoi Tessellation feature. For experiments 1, 2, 3, and 4, two SVM classifiers were built: one uses linear kernel while the other uses Gaussian kernel.

The aim of the fifth experiment is to reduce the number of features using a feature selection technique. For the fifth experiment, a pipeline was implemented. Features from experiment 4 were used. Based on the suggestion by Guyon & Elisseeff (2003), Feature selection was applied by using L1-Based Feature Selection with Linear SVM as the estimator. The features selected were used to build an SVM with Gaussian kernel. The SVM with Gaussian Kernel classifier was then evaluated. It turned out that only one feature was excluded: Average time defenders defending their respective matchups. The remaining 7 features were selected to be used as features for the SVM with Gaussian Kernel. Table 4.2 shows the scores of the SVM classifiers for each experiment.

Table 4.2. Scores for the five experiments

Experiment No	Sensitivity	Specificity	MCC	G-mean
1	0.8571	89.39	0.6706	0.8753
2	0.8571	**0.9091**	0.6972	0.8827
3	**1.0000**	0.8636	**0.7250**	**0.9293**
4	0.8571	0.8182	0.5599	0.8374
5	0.8571	0.8182	0.5599	0.8374

As seen on Table 4.2, the ability of a classifier to predict true positives is best when each team based features is used. The tradeoff, however, is that the ability to predict true negatives decreases. Classifiers built with individual based features classify man-to-better than those built with team based classifiers. Furthermore, the presence of excess area feature did not increase nor decrease the performance of the classifiers from the five models in correctly predicting zone defenses.

In terms of MCC, the models in the third experiment gave the best prediction among all experiments. As the features got smaller, the predictive performance of the model gets lower in terms of MCC. Based on the MCC obtained from the five experiments, it is safe to say that the models do not perform random nor disagreeable predictions.

4.2 Error analysis

By empirically looking at the G-mean scores, it can be observed that the performance of the classifier on both classes is at best when team based features and individual based features are combined. The model in experiment 3 yielded with the highest G-mean score. However, it is also important to note that empirically all classifiers have quite good performance all throughout.

The sampling technique called stratified k fold cross validation (with k=10) was applied upon building the model. Because of this this technique, the models produced were able to perform well in predicting examples in the test set, which they were not able to encounter upon training. Figures 4.1, 4.2, 4.3, 4.4, and 4.5 show the learning curves for each experiment with G-mean as the scoring metric.

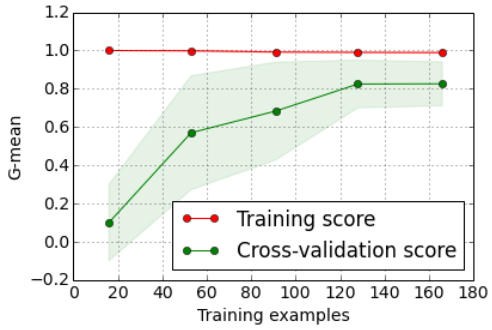


Figure 4.1. Learning Curves for Experiment 1

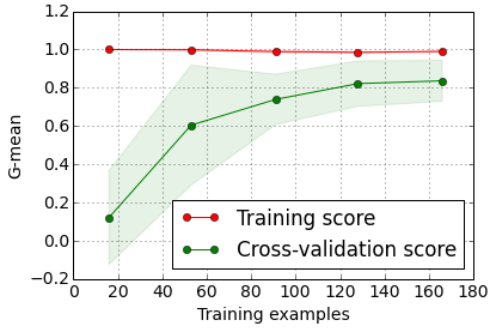


Figure 4.2. Learning Curves for Experiment 2

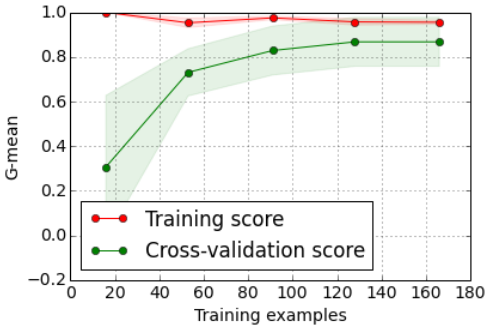


Figure 4.3. Learning Curves for Experiment 3

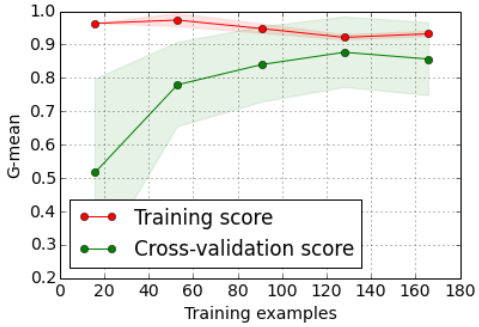


Figure 4.4. Learning Curves for Experiment 4

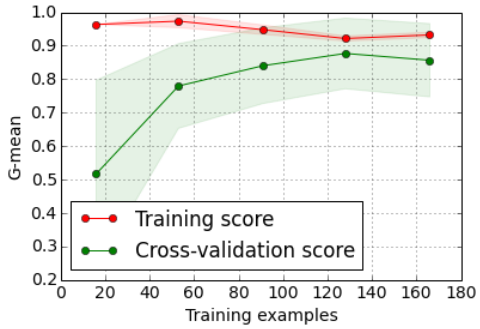


Figure 4.5. Learning Curves for Experiment 5

As shown in Figures 4.1, 4.2, 4.3, 4.4, and 4.5, the G-mean score of the cross validation set increases as the number of training examples increase. Furthermore, as shown in the figures, the models do not suffer from underfitting problem especially that the scores are high. This indicates that the model is able to learn from the data. However, the models in the first and second experiment, even with their high scores in MCC and specificity, indicate over fitting. The training scores are almost at the maximum regardless of the increase of training examples. Compared to other models, the gap between the training score and testing score for both models are somehow large.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusion

This study presents a methodology for processing raw SportVU data as inputs to the machine learning classifier. With the use of play-by-play data, the events after inbound and after timeout plays were extracted. The events were then extracted through the raw SportVU data and then segmented and converted into actions using the rule based algorithm described in Figure 3.3. Features that are relevant to defensive play classification such as the ones described in Table 3.2 can be extracted from the actions and preprocessed to become inputs for the defensive play classifier.

This study also suggests that classifying defensive play using machine learning is possible. Five classifiers were built, with each having advantages and disadvantages. The five classifiers perform generally well. However, upon analyzing the learning curves, the models from first and second experiments experienced overfitting. Hence, additional data could help in overcoming overfitting. Overall, the model from third experiment generally seems to be the best in classifying both defensive schemes despite the imbalance on the number of examples. This shows that the set of features with team based features and the individual excess Voronoi areas would lead to a good classification performance.

5.2 Recommendations for Future Work

5.2.1 More data, advice from experts

The performance of the classifiers would definitely improve if more data is added. In this study, the data was labelled by the researcher itself based on his knowledge about basketball. This poses a risk in the sense that some examples might really be misclassified due to human error. The data labelling for this study is somehow expensive in terms of time and the type of people needed who does the labelling. Despite the expense, data labelling is better done by the experts in basketball in order to get a better amount of correctly labelled data. If the dataset of this study is used again to improve the performance without adding more data, another sampling technique might be used such as bootstrapping.

5.2.2 Refine classification to more specific types of defensive plays

There are a lot of defensive schemes available in basketball. Zone defense also has a lot of variations. Zone defense and man-to-man defense can be played simultaneously: hybrid defense. There are also full court types of defenses. This study primarily focused on two types of half-court defensive schemes: 2-3 zone defense and man-to-man defense. When considering other types of defense into the classifier, some features might be added or removed, and other types of machine learning models might be used such as Recurrent Neural Network or Long Short Term Memory especially that the data contains spatiotemporal properties. This study can be used to coarsely classify or separate man-to-man defense from other types of defense.

5.2.3 Explore other feature selection techniques

Features derived from the feature selection did not really improve overall classification performance. In the fifth experiment, an embedded method was used with linear SVM as a filter, and the "filtered" or feature subset was used to train a nonlinear SVM (using RBF kernel). However, Based on the G-mean and MCC scores of the model of the fifth experiment was lower than the first four experiments. Other feature selection techniques such as recursive feature elimination on an SVM with a linear kernel, genetic algorithm, or use a decision tree based classifier to get information on the importance of the features. Features extracted from the SportVU data was purely knowledge based. Feature extraction for this study remains as an area to be explored. Other features might be extracted and would be able to separate the classes better than what is presented in this study.

5.2.4 Data cleaning

Further cleaning of the SportVU data is needed. Some frames of the SportVU data does not really project what happened in the game. At some frames, some players were not detected. Hence the events containing these kinds of frames were not considered in this study. Some events were not really synchronized with the play-by-play data. For example, the event logged in the play-by-play data ends at 0:21 but in the SportVU data the event ends 2 to 3 seconds earlier or later. Thus, some unnecessary player trajectories would be extracted, and also some important player trajectories would not be extracted.

APPENDIX A

Sample Box Score data (National Basketball Association, 2015)

SHARE ON: http://on.nba.com/2kd

Detroit Pistons

PLAYER	MIN	FGM	FGA	FG%	3PM	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST	TOV	STL	BLK	PF	PTS	+/-
Marcus Morris F	57:17	7	17	41.2	3	6	50.0	3	6	50.0	2	5	7	2	2	0	1	6	20	7
Ersan Ilyasova F	53:57	6	17	35.3	1	8	12.5	5	8	62.5	3	6	9	4	2	0	0	5	18	-3
Andre Drummond C	54:12	14	25	56.0	0	0	0.0	5	10	50.0	10	11	21	3	2	3	2	6	33	-3
Kentavious Caldwell-…	48:45	7	18	38.9	2	7	28.6	1	4	25.0	1	6	7	3	1	1	1	5	17	-6
Reggie Jackson G	49:30	12	27	44.4	0	2	0.0	7	10	70.0	1	5	6	13	2	1	0	4	31	-1
Steve Blake	19:22	3	5	60.0	1	2	50.0	1	2	50.0	1	2	3	2	0	1	0	0	8	3
Stanley Johnson	30:24	6	11	54.5	1	2	50.0	3	4	75.0	1	6	7	1	2	1	0	6	16	6
Anthony Tolliver	12:58	1	4	25.0	0	2	0.0	0	0	0.0	0	1	1	0	0	0	0	1	2	5
Joel Anthony	13:35	0	0	0.0	0	0	0.0	2	2	100	0	3	3	0	0	0	0	2	2	7
Aron Baynes	DNP - COACH'S DECISION																			
Reggie Bullock	DNP - COACH'S DECISION																			
Spencer Dinwiddie	DNP - COACH'S DECISION																			
Darrun Hilliard	DNP - COACH'S DECISION																			
Totals:		56	124	45.2	8	29	27.6	27	46	58.7	19	45	64	28	11	7	4	35	147	3

Chicago Bulls

PLAYER	MIN	FGM	FGA	FG%	3PM	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST	TOV	STL	BLK	PF	PTS	+/-
Tony Snell F	38:11	1	7	14.3	1	7	14.3	0	0	0.0	1	3	4	0	2	1	0	5	3	9
Taj Gibson F	44:06	4	7	57.1	0	0	0.0	6	7	85.7	6	6	12	0	2	0	2	2	14	-2
Pau Gasol C	48:03	10	23	43.5	0	1	0.0	10	12	83.3	6	9	15	5	1	0	5	5	30	9
Jimmy Butler G	55:33	14	29	48.3	1	4	25.0	14	16	87.5	1	7	8	2	3	2	2	5	43	-5
Derrick Rose G	54:13	14	34	41.2	1	3	33.3	5	5	100	0	4	4	8	4	1	0	5	34	5
Doug McDermott	22:19	1	3	33.3	0	2	0.0	0	0	0.0	0	1	1	0	0	0	0	3	2	-10
Joakim Noah	25:40	3	5	60.0	0	0	0.0	0	0	0.0	5	6	11	4	3	1	0	1	6	-10
Nikola Mirotic	23:56	1	6	16.7	1	3	33.3	2	2	100	0	5	5	0	0	0	0	3	5	-7
Aaron Brooks	11:30	1	3	33.3	0	0	0.0	0	0	0.0	1	1	2	2	1	0	0	2	2	-6
E'Twaun Moore	3:21	1	2	50.0	1	1	100	2	2	100	0	0	0	0	0	0	0	1	5	0
Kirk Hinrich	13:08	0	1	0.0	0	1	0.0	0	0	0.0	0	0	0	0	0	0	0	3	0	2
Cameron Bairstow	DNP - COACH'S DECISION																			
Bobby Portis	DNP - COACH'S DECISION																			
Totals:		50	120	41.7	5	22	22.7	39	44	88.6	20	42	62	21	16	5	9	35	144	-3

MIN — Minutes Played
FGM — Field Goals Made
FGA — Field Goals Attempted
FG% — Field Goal Percentage
3PM — Three Pointers Made
3PA — Three Pointers Attempted
3P% — Three Point Percentage
FTM — Free Throws Made
FTA — Free Throws Attempted
FT% — Free Throw Percentage
OREB — Offensive Rebounds
DREB — Defensive Rebounds
REB — Rebounds
AST — Assists
TOV — Turnovers
STL — Steals
BLK — Blocked Shots
PF — Personal Foul
PTS — Points
+/- — Plus-Minus

INACTIVE PLAYERS
DET: Brandon Jennings, Jodie Meeks
CHI: Mike Dunleavy, Cristiano Felicio

APPENDIX B

Sample Play by play data (National Basketball Association, 2015)

Detroit Pistons		Chicago Bulls
	Start of Q1	
	12:00	Jump Ball Gasol vs. Drummond: Tip to Rose
Caldwell-Pope STEAL (1 STL)	11:46	Rose Bad Pass Turnover (P1.T1)
	11:29	Snell P.FOUL (P1.T1) (J.Capers)
MISS Marc Morris 19' Jump Shot	11:19	
	11:17	Butler REBOUND (Off:0 Def:1)
	11:11	MISS Rose 1' Running Layup
Drummond REBOUND (Off:0 Def:1)	11:08	
Drummond 4' Jump Shot (2 PTS)	10:57 **2 - 0**	
	10:38	MISS Rose 2' Layup
Drummond REBOUND (Off:0 Def:2)	10:37	
Caldwell-Pope 1' Layup (2 PTS) (Jackson 1 AST)	10:29 **4 - 0**	
	10:17 **4 - 2**	**Gibson 14' Jump Shot (2 PTS) (Butler 1 AST)**
Caldwell-Pope Cutting Dunk Shot (4 PTS) (Jackson 2 AST)	10:06 **6 - 2**	
	9:51 **6 - 4**	**Gasol 10' Hook Shot (2 PTS)**
	9:32	Snell S.FOUL (P2.T2) (J.Capers)
MISS Caldwell-Pope Free Throw 1 of 2	9:32	
Pistons Rebound	9:32	
	9:32	SUB: McDermott FOR Snell
Caldwell-Pope Free Throw 2 of 2 (5 PTS)	9:32 **7 - 4**	
	9:15	MISS Gasol 19' Jump Shot
Caldwell-Pope REBOUND (Off:0 Def:1)	9:14	
Drummond 4' Hook Shot (4 PTS) (Caldwell-Pope 1 AST)	9:05 **9 - 4**	
Ilyasova P.FOUL (P1.T1) (J.Capers)	8:56	
	8:41	MISS Gasol 2' Driving Layup
Ilyasova REBOUND (Off:0 Def:1)	8:40	
MISS Drummond 2' Layup	8:29	Gasol BLOCK (1 BLK)
	8:27	McDermott REBOUND (Off:0 Def:1)
Jackson P.FOUL (P1.T2) (G.Zielinski)	8:24	
	8:17 **9 - 6**	**Gasol 14' Jump Shot (4 PTS) (Rose 1 AST)**
Ilyasova 19' Jump Shot (2 PTS) (Jackson 3 AST)	7:54 **11 - 6**	

APPENDIX C

SportVU JSON structure

- gameid (code matches)
- gamedate (date matches)
- ***events (list)***
  - eventId (code of the event)
  - home (details of the home team)
    - ***players (list)***
      - playerid (code player)
      - firstname (name)
      - lastname (surname)
      - jersey (jersey)
      - position (gaming site)
    - teamid (code teams)
    - name (name of team)
    - abbreviation (abbreviation of the name)
  - visitor (data of the away team)
  - ***moments (list of moments)***
    - [0] quarter (quarter)
    - [1] time (Time in milliseconds)
    - [2] gametime (playing time quarters)
    - [3] shottime (Time Attack)
    - [5] positions (list of positions of the ball and players (10 + 1 elements list))
      - [0] teamid (code teams)
      - [1] playerid (code player)
      - [2] x
      - [3] y
      - [4] z (height of the ball)

APPENDIX D

NBA play by play JSON structure

- resource
- parameters
  - GameID
  - StartPeriod
  - EndPeriod
- resultSets
  - name
  - headers (list of details)
  - rowSet (set of play-by-play stats based on headers)
    [0]GAMEID
    [1]EVENTNUM
    [2]EVENTMSGTYPE
    [3]EVENTMSGACTIONTYPE
    [4]PERIOD
    [5]WCTIMESTRING
    [6]PCTIMESTRING
    [7]HOMEDESCRIPTION
    [8]NEUTRALDESCRIPTION
    [9]VISITORDESCRIPTION
    [10]SCORE
    [11]SCOREMARGIN
    [12]PERSON1TYPE
    [13]PLAYER1_ID
    [14]PLAYER1_NAME
    [15]PLAYER1_TEAM_ID
    [16]PLAYER1_TEAM_CITY
    [17]PLAYER1_TEAM_NICKNAME
    [18]PLAYER1_TEAM_ABBREVIATION
    [19]PERSON2TYPE
    [20]PLAYER2_ID
    [21]PLAYER2_NAME
    [22]PLAYER2_TEAM_ID
    [23]PLAYER2_TEAM_CITY
    [24]PLAYER2_TEAM_NICKNAME
    [25]PLAYER2_TEAM_ABBREVIATION
    [26]PERSON3TYPE
    [27]PLAYER3_ID
    [28]PLAYER3_NAME
    [29]PLAYER3_TEAM_ID
    [30]PLAYER3_TEAM_CITY
    [31]PLAYER3_TEAM_NICKNAME
    [32]PLAYER3_TEAM_ABBREVIATION

APPENDIX E

Code. Full Code seen on https://github.com/rjesteban

Action Code

```
import json

PATH = 'data/actions/'


def load_action(gameid, eid):
   with open(PATH + str(gameid) + '.json') as f:
       data = json.load(f)
   ctx = data[str(eid)]
   action = Action(gameid=str(ctx['gameid']), eid=ctx['eventid'],
                   coords=ctx['coords'], time=ctx['time'],
                   quarter=ctx['quarter'], offense=ctx['offense'],
                   defense=ctx['defense'], label=ctx['label'])
   return action


class Action(object):
   def __init__(self, gameid=None, eid=None, coords=None,
                time=None, quarter=None,
                offense=None, defense=None, label=None):
       self.eventid = eid
       self.coords = coords
       self.time = time
       self.quarter = quarter
       self.offense = offense
       self.defense = defense
       self.gameid = gameid
       self.label = label

   def save(self):
       context = {}
       context['eventid'] = self.eventid
       context['coords'] = self.coords
       context['time'] = self.time
       context['quarter'] = self.quarter
       context['offense'] = self.offense
       context['defense'] = self.defense
       context['gameid'] = self.gameid
       context['label'] = self.label
       try:
           with open(PATH + str(self.gameid) + '.json', 'r') as f:
               data = json.load(f)
           with open(PATH + str(self.gameid) + '.json', 'w') as f:
               data[str(context['eventid'])] = context
               json.dump(data, f)
       except Exception:
           with open(PATH + str(self.gameid) + '.json', 'w') as f:
               json.dump({str(context['eventid']): context}, f)
       return context
```

Features

```
def get_entropy(action):
    """
    Gets the entropy of each defender
    Parameters
    ----------
    action : Action instance
    Returns
    ----------
    entropy : list of entropy
    """
    entropy = sum(determine_matchup_over_time(action))  # [25:-25])
    length = len(action.coords)  # [25:-25])
    for r in range(len(entropy)):  # defenders
        for c in range(len(entropy[0])):  # offenders
            if entropy[r][c] > 0:
                p = entropy[r][c] / float(length)
                entropy[r][c] = -(p * math.log(p, 2))
    return [sum(row) for row in entropy]


def get_mean_distance(action):
    """
    Gets the mean distance of each defender from its offender
    Parameters
    ----------
    action : Action instance
    Returns
    ----------
    mean_distance : list of mean distance
    """
    matchup = determine_matchup_over_time(action)
    length = len(matchup)
    match = [p.argmax() for p in matchup[25]]  # not sum(matchup)
    dist = [0 for row in range(5)]
    for time in range(len(matchup)):
        defenders = get_coords(action.coords[time], action.defense)
        offenders = get_coords(action.coords[time], action.offense)
        for d in range(5):  # defenders
            dist[d] += get_distance(defenders[d], offenders[match[d]])
    return [float(p) / length for p in dist]


def get_mean_distance_from_canonical_position(action):
    """
    Gets the mean distance of each defender from its cannonical position
    Parameters
    ----------
    action : Action instance
    Returns
    ----------
    entropy : list of mean distance from canonical position
    """
    matchup = determine_matchup_over_time(action)
    length = len(matchup)
    match = [p.argmax() for p in matchup[25]]  # sum(matchup)
    dist = [0 for r in range(5)]
```

```
    for time in range(len(matchup)):
        defenders = get_coords(action.coords[time], action.defense)
        canon = get_canonical_position(action, time)
        for d in range(5):  # defenders
            dist[d] += get_distance(defenders[d], canon[match[d]])
    return [float(p) / length for p in dist]


def get_time_defending(action):
    """
    Gets the time a defender defends its matchup at frame 25
    Parameters
    ----------
    action : Action instance
    Returns
    ----------
    time : list of time defending (frame based)
    """
    matchup = determine_matchup_over_time(action)
    match = [p.argmax() for p in matchup[25]]  # sum(matchup)
    defending = get_mean_distance_from_canonical_position(action)
    return [float(defending[d] *
            float(sum(matchup[25:-25])[index][d] / len(matchup)))
            for index, d in enumerate(match)]


def get_distance_from_post(action):
    length = len(action.coords)
    arrange_by_position(action)
    defenders = [get_coords(action.coords[time], action.defense)
                 for time in range(length)][0:-25]
    post = defenders[0]
    dist_from_post = [0 for r in range(5)]
    for coords in defenders:
        for d in range(5):
            dist_from_post[d] += get_distance(coords[d], post[d])
    return [float(p) / len(defenders) for p in dist_from_post]


def get_diff_canon_vs_matchup(action):
    matchup = determine_matchup_over_time(action)
    length = len(matchup)
    match = [p.argmax() for p in matchup[25]]  # sum(matchup)
    dist = [0 for r in range(5)]
    for time in range(len(matchup)):
        defenders = get_coords(action.coords[time], action.defense)
        canon = get_canonical_position(action, time)
        offenders = get_coords(action.coords[time], action.offense)
        for d in range(5):  # defenders
            dist[d] += (get_distance(defenders[d], offenders[match[d]]) -
                        get_distance(canon[match[d]], offenders[match[d]]))
    return [float(p) / length for p in dist]


# average speed = distance moved / time taken
def get_average_speed_defense(action):
    length = len(action.coords)
    arrange_by_position(action)
```

```
    defenders = [get_coords(action.coords[time], action.defense)
                 for time in range(length)][0:-25]
    dist = [0 for r in range(5)]
    for index, coords in enumerate(defenders):
        for d in range(5):
            dist[d] += get_distance(coords[d], defenders[index - 1][d])
    return [float(p) / len(defenders) for p in dist]


# average speed = distance moved / time taken
def get_average_speed_offense(action):
    length = len(action.coords)
    arrange_by_position(action)
    offenders = [get_coords(action.coords[time], action.offense)
                 for time in range(length)][0:-25]
    dist = [0 for r in range(5)]
    for index, coords in enumerate(offenders):
        for d in range(5):
            dist[d] += get_distance(coords[d], offenders[index - 1][d])
    return [float(p) / len(offenders) for p in dist]


def get_excess_voronoi_area(action):
    length = len(action.coords)
    arrange_by_position(action)
    defenders = [get_coords(action.coords[time], action.defense)
                 for time in range(length)][25:-25]
    ref_areas = get_voronoi_areas(defenders[0])
    avg_areas = [0 for r in range(5)]
    for points in defenders:
        areas = get_voronoi_areas(points)
        for i in range(5):
            avg_areas[i] += areas[i]

    return ([ref_areas[p] - (avg / len(defenders))
            for p, avg in enumerate(avg_areas)])

def voronoi_finite_polygons_2d(vor, radius=None):
    """
    Reconstruct infinite voronoi regions in a 2D diagram to finite
    regions.
    Parameters
    ----------
    vor : Voronoi
        Input diagram
    radius : float, optional
        Distance to 'points at infinity'.
    Returns
    -------
    regions : list of tuples
        Indices of vertices in each revised Voronoi regions.
    vertices : list of tuples
        Coordinates for revised Voronoi vertices. Same as coordinates
        of input vertices, with 'points at infinity' appended to the
        end.
    """


    if vor.points.shape[1] != 2:
```

```
    raise ValueError("Requires 2D input")

new_regions = []
new_vertices = vor.vertices.tolist()

center = vor.points.mean(axis=0)
if radius is None:
    radius = vor.points.ptp().max() * 2
    print "radius: " + str(radius)

# Construct a map containing all ridges for a given point
all_ridges = {}
for (p1, p2), (v1, v2) in zip(vor.ridge_points, vor.ridge_vertices):
    all_ridges.setdefault(p1, []).append((p2, v1, v2))
    all_ridges.setdefault(p2, []).append((p1, v1, v2))

# Reconstruct infinite regions
for p1, region in enumerate(vor.point_region):
    vertices = vor.regions[region]

    if all(v >= 0 for v in vertices):
        # finite region
        new_regions.append(vertices)
        continue

    # reconstruct a non-finite region
    ridges = all_ridges[p1]
    new_region = [v for v in vertices if v >= 0]

    for p2, v1, v2 in ridges:
        if v2 < 0:
            v1, v2 = v2, v1
        if v1 >= 0:
            # finite ridge: already in the region
            continue

        # Compute the missing endpoint of an infinite ridge

        t = vor.points[p2] - vor.points[p1]  # tangent
        t /= np.linalg.norm(t)
        n = np.array([-t[1], t[0]])  # normal

        midpoint = vor.points[[p1, p2]].mean(axis=0)
        direction = np.sign(np.dot(midpoint - center, n)) * n
        far_point = vor.vertices[v2] + direction * radius

        new_region.append(len(new_vertices))
        new_vertices.append(far_point.tolist())

    # sort region counterclockwise
    vs = np.asarray([new_vertices[v] for v in new_region])
    c = vs.mean(axis=0)
    angles = np.arctan2(vs[:, 1] - c[1], vs[:, 0] - c[0])
    new_region = np.array(new_region)[np.argsort(angles)]

    # finish
```

```
        new_regions.append(new_region.tolist())

    return new_regions, np.asarray(new_vertices)


def get_voronoi_areas(points):
    points = np.array(points)
    min_x, max_x, min_y, max_y = -5.0, 52.0, -5.0, 55.0
    vor = Voronoi(points)
    regions, vertices = voronoi_finite_polygons_2d(vor, 1500)
    box = Polygon([[min_x, min_y], [min_x, max_y], [max_x, max_y],
                   [max_x, min_y]])
    areas = [-1 for num in range(5)]
    for region in regions:
        polygon = vertices[region]
        # Clipping polygon
        poly = Polygon(polygon).intersection(box)

        for i, (x, y) in enumerate(points):
            point = Point(x, y)
            if point.within(poly):
                areas[i] = poly.area
                break

        # polygon = [p for p in poly.exterior.coords]
        # plt.fill(*zip(*polygon), alpha=0.4)

    # plt.plot(points[:, 0], points[:, 1], 'ko')
    # plt.axis('equal')
    # plt.xlim(-5, 52)
    # plt.ylim(-4, 55)
    # plt.show()

    for i, area in enumerate(areas):
        if area <= 0:
            raise Exception("Point not assigned to a Voronoi Cell: " +
                            str(i) + " | " + str(points))

    return areas
```

Preprocessor Codes

```
import math
import numpy as np


def determine_matchup_over_time(action):
    matchup = []
    length = len(action.coords)
    for i in range(length):
        matchup.append(determine_matchup(action, i, "canonical"))
    return matchup
```

```
def get_distance(p1, p2):
    sidex = math.fabs(p1[0] - p2[0])
    sidey = math.fabs(p1[1] - p2[1])
    return math.fabs(math.sqrt(sidex ** 2 + sidey ** 2))


def transform_wlog(action):
    coordinates = []
    for entity in action.coords:
        for e in entity:
            x = e[2]
            y = e[3]
            if x > 47:
                e[2] = round(94 - x, 4)
            else:
                e[3] = round(50 - y, 4)
        coordinates.append(entity)
    action.coords = coordinates
    return action


def arrange_by_position(action):
    rank = {'C': 0, 'C-F': 1, 'F-C': 2, 'F': 3, 'F-G': 4, 'G-F': 5, 'G': 6}
    offense = range(len(action.offense))
    for i in range(len(action.offense)):
        j = i
        key = rank[action.offense[i][1]]
        while j > 0 and rank[str(offense[j - 1][1])] > key:
            offense[j] = offense[j - 1]
            j -= 1
        offense[j] = action.offense[i]

    defense = range(len(action.defense))
    for i in range(len(action.defense)):
        j = i
        key = rank[action.defense[i][1]]
        while j > 0 and rank[str(defense[j - 1][1])] > key:
            defense[j] = defense[j - 1]
            j -= 1
        defense[j] = action.defense[i]
    action.offense = offense
    action.defense = defense
    return action


def get_coords(coords, players):
    """
    returns : array of [x, y] coordinates according to order of players.
    """
    ctk = []
    for p in players:
        for coord in coords:
            pid = coord[1]
            if str(pid) == str(p[0]):
                ctk.append([coord[2], coord[3]])
    return ctk
```

```
def get_canonical_position(action, time):
    arrange_by_position(action)
    ball = (action.coords[time][0][2], action.coords[time][0][3])
    otk = get_coords(action.coords[time], action.offense)
    hoop = (4.0, 25.0)
    pos = []
    for o in otk:
        xpos = (0.62 * o[0]) + (0.11 * ball[0]) + (0.27 * hoop[0])
        ypos = (0.62 * o[1]) + (0.11 * ball[1]) + (0.27 * hoop[1])
        pos.append([xpos, ypos])
    return pos


def determine_matchup(action, time, by):
    matrix = np.zeros((5, 5), dtype=np.float32)
    arrange_by_position(action)
    defenders = get_coords(action.coords[time], action.defense)
    if by == "distance":
        offender_loc = get_coords(action.coords[time], action.offense)
    else:
        offender_loc = get_canonical_position(action, time)

    # defender rows
    # offender cols
    for i, (px, py) in enumerate(defenders):
        a = [get_distance((px, py), (d[0], d[1])) for d in offender_loc]
        matrix[i][a.index(min(a))] = 1
    return matrix
```

Acquire Play by play data code

```
import json
import os
import requests

PATH = 'data/pbp/'


def acquire_pbp_json(id):
    header_data = {
        'Accept-Encoding': 'gzip, deflate, sdch',
        'Accept-Language': 'en-US,en;q=0.8',
        'Upgrade-Insecure-Requests': '1',
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64)'
        ' AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.82 '
        'Safari/537.36',
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9'
        ',image/webp,*/*;q=0.8',
        'Cache-Control': 'max-age=0',
        'Connection': 'keep-alive'
    }
    game_url = ('http://stats.nba.com/stats/playbyplayv2?' +
                'EndPeriod=0&EndRange=0&GameID=' + id +
                '&RangeType=0&StartPeriod=0&StartRange=0')
    response = requests.get(game_url, headers=header_data)
    data = response.json()
    if not os.path.exists(os.path.dirname(PATH)):
```

```
        os.makedirs('pbp')
    with open(PATH + id + 'pbp.json', 'w') as file:
        json.dump(data, file)


def get_pbp(id):
    return json.load(open(PATH + id + 'pbp.json'))


def find_index(data, eid):
    for index, row in enumerate(data['resultSets'][0]['rowSet']):
        if str(int(row[1])) == str(int(eid)):
            return index
    raise Exception("Index not found")


def get_play(gid, eid):
    data = get_pbp(gid)
    index = find_index(data, eid)
    return data['resultSets'][0]['rowSet'][index]
```

Segmenter Code

```
import math

PBP_PATH = 'data/pbp/'


def time_difference(time1, time2):
    time_1 = [int(i) for i in time1.split(':')]
    time_2 = [int(i) for i in time2.split(':')]
    t1 = (time_1[0] * 60) + time_1[1]
    t2 = (time_2[0] * 60) + time_2[1]
    if t1 < t2:
        raise Exception(time1 + " is less than " + time2)
    return int(math.fabs(t1 - t2))


def all_on_one_side(m, eid, frame):
    return len(set([0 if coord[2] < 47 else 1 for coord in m[frame][5]])) == 1


def within_the_paint(entity, eid):
    x, y = entity[2], entity[3]
    return (0 <= x <= 19 or (94 - 19) <= x <= 94 and
            (25 - 12) <= y <= (25 + 12))


# min1 sec1 less than min2 step2
def less_than(min1, sec1, min2, step2):
    if min1 < min2:
        return True
    return min1 == min2 and sec1 < step2
```

```
def format_time(time):
   if isinstance(time, list):
       time = 1
   mins = int(time / 60)
   secs = int(((time / 60.0) - mins) * 60)
   return str(mins).zfill(2) + ':' + str(secs).zfill(2)

from action.core import Action
from segmenter.utils import (time_difference, all_on_one_side,
                             within_the_paint, less_than, format_time)
from preprocessor.utils import transform_wlog
from playbyplay.utils import get_play
from sportvu.utils import get_moment, determine_offs_defs
import json

PBP_PATH = 'data/pbp/'


# time difference quota is still under experiment, EDA needed
def pick_possessions(gameid):
   """
   picks ids of moments ending in a shot, or a steal.
   do not pick moments where the previous moment was a steal.
   pick after timeout/violation/substitution/foul moments.

   Keyword argument:
   gameid -- file name (str)

   returns:
       array of gameid, eventid
   """
   data = json.load(open(PBP_PATH + gameid + 'pbp.json'))
   row_set = data['resultSets'][0]['rowSet']
   moments = []

   for i in range(1, len(row_set)):
       if ((row_set[i][7] is not None and 'S.FOUL' in row_set[i][7]) or
          (row_set[i][9] is not None and 'S.FOUL' in row_set[i][9])):
           shot_foul = True
       else:
           shot_foul = False
       shot_attempt = ((1 <= row_set[i][2] <= 2) or shot_foul)
       turnover = row_set[i][2] == 5
       # stop_play = row_set[i][2] in [7, 9] and
       # row_set[i - 1][2] not in [7, 9]
       if ((row_set[i][7] is not None and 'Timeout' in row_set[i][7]) or
          (row_set[i][9] is not None and 'Timeout' in row_set[i][9]) or
          (row_set[i][7] is None and row_set[i][9] is None)):
           timeout = True
       else:
           timeout = False
       came_from_steal = (row_set[i - 1][2] == 5 and
                          row_set[i - 1][3] <= 2 and
                          time_difference(row_set[i - 1][6],
                          row_set[i][6]) < 5)
       follow_up = (row_set[i - 1][2] == 4 and
                    time_difference(row_set[i - 1][6], row_set[i][6]) < 5)
       attempt = (shot_attempt and not follow_up) or turnover  # or stop_play
```

```
        if attempt and not came_from_steal and not timeout:
            moments.append(row_set[i][1])
    return moments


# Rule based algorithm
def convert_moment_to_action(data, eid, check_frames=True, label=0):
    play = get_play(str(data['gameid']), eid)
    end_time = play[6]
    end_min = int(end_time.split(':')[0])
    end_sec = int(end_time.split(':')[1])
    moment = get_moment(data, eid)
    gameid = str(data['gameid'])
    frames = []
    prev_frames = []
    inside_count = 0
    for fr, frame in enumerate(moment):
        ball = moment[fr][5][0]
        mins = int(moment[fr][2] / 60)
        secs = int(((moment[fr][2] / 60.0) - mins) * 60)
        if less_than(mins, secs, end_min, end_sec):
            break
        if all_on_one_side(moment, eid, fr):
            frames.append(frame)
            if within_the_paint(ball, eid):
                inside_count += 1
            else:
                inside_count = 0
            # height of ball: ball[4]
            # if inside_count > 50 or ball[4] > 12 and len(frames) >= 120:
            if ball[4] > 13 and len(frames) >= 130:
                prev_frames = frames
                frames = []
                inside_count = 0
        else:
            if len(frames) >= 130:
                prev_frames = frames
            inside_count = 0
            frames = []
    if check_frames and (len(frames) < 130):
        if len(prev_frames) >= 130:
            frames = prev_frames
        else:
            raise Exception("Insufficient number of frames: " +
                            str(len(frames)) + " | " + str(len(prev_frames)))
    players = determine_offs_defs(data, gameid, eid)
    offense = players['offense']
    defense = players['defense']
    coords = [coord[5] for coord in frames]
    quarter = frames[0][0]
    the_time = [format_time(time[2]) for time in frames]
    action = Action(gameid=gameid, eid=eid, coords=coords,
                    time=the_time, quarter=quarter,
                    offense=offense, defense=defense, label=label)
    transform_wlog(action)
    return action
```

REFERENCES

Accenture. (2016). *Intelligent Automation: The essential new co-worker for the digital age*. Accenture. Retrieved from https://www.accenture.com/t20160125T111718__w__/us-en/_acnmedia/Accenture/Omobono/TechnologyVision/pdf/Intelligent-Automation-Technology-Vision-2016.pdf

Bi, J., Bennett, K., Embrechts, M., Breneman, C., & Song, M. (2003). Dimensionality Reduction via Sparse Support Vector Machines. *Journal Of Machine Learning Research*, *3*, 1229-1243. Retrieved from http://www.jmlr.org/papers/volume3/bi03a/bi03a.pdf

Big Data meets big-time basketball (2014, May 22) retrieved 5 October 2016 from http://phys.org/news/2014-05-big-big-time-basketball.html

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Cervone, D., D'Amour, A., Bornn, L., & Goldsberry, K. (2014). POINTWISE: Predicting Points and Valuing Decisions in Real Time with NBA Optical Tracking Data. In *8th Annual MIT Sloan Sports Analytics Conference*. Boston. Retrieved from http://www.sloansportsconference.com/wp-content/uploads/2014/02/2014_SSAC_Pointwise-Predicting-Points-and-Valuing-Decisions-in-Real-Time.pdf

Bekkar, M., Djemaa, H., & Alitouche, T. (2013). Evaluation Measures for Models Assessment over Imbalanced Data Sets. *Journal of Information Engineering and Applications*, 3(10), 27-38. Retrieved from http://www.iiste.org/Journals/index.php/JIEA/article/view/7633

Duda, R. O., Hart, P. E. 1., & Stork, D. G. (2006). *Pattern classification* (2nd ed.). New York ; New Delhi: Wiley.

Franks, I. & Miller, G. (1986). Eyewitness testimony in sport. *Journal Of Sport Behavior*, *9*(1), 38-45. Retrieved from http://psycnet.apa.org/psycinfo/1987-10213-001

Franks, A., Miller, A., Bornn, L., & Goldsberry, K. (2015a). Characterizing the spatial structure of defensive skill in professional basketball. *The Annals Of Applied Statistics*, *9*(1), 94-121. http://dx.doi.org/10.1214/14-aoas799. Retrieved from https://arxiv.org/pdf/1405.0231v2.pdf

Franks, A., Miller, A., Bornn, L., & Goldsberry, K. (2015b). Counterpoints: Advanced Defensive Metrics for NBA Basketball. In *9th Annual MIT Sloan Sports Analytics Conference*. Boston. Retrieved from http://www.sloansportsconference.com/wp-content/uploads/2015/02/SSAC15-RP-Finalist-Counterpoints2.pdf

Gudmundsson, J., & Horton, M. (2017). Spatio-Temporal Analysis of Team Sports. *ACM Computing Surveys*, 50(2), 1-34. http://dx.doi.org/10.1145/3054132

Goldsberry, K. & Weiss, E. (2013). The Dwight Effect: A New Ensemble of Interior Defense Analytics for the NBA. In *7th Annual MIT Sloan Sports Analytics Conference*. Boston Convention and Exhibition Center, MA, USA. Retrieved from http://www.sloansportsconference.com/wp-content/uploads/2013/The%20Dwight%20Effect%20A%20New%20Ensemble%20of%20Interior%20Defense%20Analytics%20for%20the%20NBA.pdf

Green, M. W., National Institute of Justice (U.S.), Safe and Drug-Free Schools Program (U.S.), & Sandia National Laboratories. (1999). *The appropriate and effective use of security technologies in U.S. schools: A guide for schools and law enforcement agencies*. Washington, DC: U.S. Dept. of Justice, Office of Justice Programs, National Institute of Justice. Retrieved from https://www.ncjrs.gov/school/178265.pdf

Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal Of Machine Learning Research*, *3*, 1157-1182. Retrieved from http://jmlr.csail.mit.edu/papers/volume3/guyon03a/guyon03a.pdf

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: concepts and techniques (3rd ed)*. Waltham, MA: Morgan Kaufmann Publishers.

Hoang, G., Bouzerdoum, A., & Lam, S. (2009). Learning Pattern Classification Tasks with Imbalanced Data Sets. *Pattern Recognition*, 193-298. http://dx.doi.org/10.5772/7544

International Basketball Federation,. *Basketball Glossary | FIBA Europe*. *Fibaeurope.com*. Retrieved 11 December 2016, from http://www.fibaeurope.com/cid_UD-XflK3IQgl4t8JKzEA00.html

International Basketball Federation. (2014). *Official Basketball Rules 2014*. Retrieved from http://www.fiba.com/documents/2015/Official_Basketball_Rules_2014_Y.pdf

International Business Machines. (2015). *Turn video into public safety insight with IBM Intelligent Video Analytics*. *http://www.ibmbigdatahub.com/*. Retrieved 6 October 2016, from http://www.ibmbigdatahub.com/video/turn-video-public-safety-insight-ibm-intelligent-video-analytics

Johnson, N. (2016). *2016 NBA Raw SportVU Game Logs*. Retrieved from https://github.com/neilmj/BasketballData/tree/master/2016.NBA.Raw.SportVU.Game.Logs

Kim, S. (2004). Voronoi analysis of a soccer game. *Nonlinear Analysis: Modelling and Control*, 9, 233–240. Retrieved 2 May 2017 from https://www.mii.lt/na/issues/NA_0903/NA09303.pdf

Kempe, M., Grunz, A., & Memmert, D. (2014). Detecting tactical patterns in basketball: Comparison of merge self-organising maps and dynamic controlled neural networks. *European Journal Of Sport Science*, *15*(4), 249-255. http://dx.doi.org/10.1080/17461391.2014.933882. Retrieved from https://www.dshs-koeln.de/fileadmin/redaktion/Institute/Kognitions-_und_Sportspielforschung/Publikationen/Paper/EJSS_2014_Kempe_et_al_PlayDetection.pdf

Lopes, A., Fonseca, S., Lese R., Baca, A., (2015). Using Voronoi diagrams to describe tactical behaviour in invasive team sports: an application in basketball. *Cuadernos de Psicología del Deporte*, 15() 123-129. Retrieved from http://www.redalyc.org/articulo.oa?id=227038699012

Mac, C. (2014). *501 Awesome Basketball Quotes | Basketball For Coaches*. *Basketballforcoaches.com*. Retrieved 21 September 2016, from htoutp://www.basketballforcoaches.com/basketball-quotes/

Maheswaran, R., Chang, Y., Henehan, A., & Danesis, S. (2012). Deconstructing the Rebound with Optical Tracking Data. In *6th Annual MIT Sloan Sports Analytics Conference*. Boston, MA, USA. Retrieved from http://www.sloansportsconference.com/wp-content/uploads/2012/02/108-sloan-sports-2012-maheswaran-chang_updated.pdf

Maymin, A., Maymin, P., & Shen, E. (2012). Individual Factors of Successful Free Throw Shooting. *Journal Of Quantitative Analysis In Sports, Forthcoming*, *8*(3). http://dx.doi.org/10.2139/ssrn.1947166

McIntyre, A., Brooks, J., Guttag, J., & Wiens, J. (2016). Recognizing and Analyzing Ball Screen Defense in the NBA. In *10th MIT Sloan Sports Analytics Conference*. Boston. Retrieved from http://www.sloansportsconference.com/wp-content/uploads/2016/02/1530-Basketball.pdf

National Basketball Association. (2001). *NBA.com - How to Read a Box Score*. *Nba.com*. Retrieved 22 September 2016, from http://www.nba.com/analysis/00422972.html

National Basketball Association. (2013a). *NBA partners with Stats LLC for tracking technology*. *NBA.com* [Official release]. Retrieved 21 September 2016, from http://www.nba.com/2013/news/09/05/nba-stats-llc-player-tracking-technology/

National Basketball Association. (2013b). *Official Rules of the National Basketball Association 2013-2014* (1st ed., p. 8). Retrieved from http://www.nba.com/media/dleague/1314-nba-rule-book.pdf

National Basketball Association. (2015). *NBA.com/Stats - DET vs CHI*. *Stats.nba.com*. Retrieved 27 December 2016, from http://stats.nba.com/game/#!/0021500391/playbyplay/

National Basketball Association. (2016). *NBA Champs: Cavs Celebration, LBJ Postgame and Trophy Presentation*. Retrieved from https://www.youtube.com/watch?v=R4CIR_nzMZA

NEC Corporation. (2012). *IFSEC_NEC 2012 - Birmingham*. Retrieved from https://www.youtube.com/watch?v=N8vDjt7dzpQ

Parker, C. (2015). *How the NBA learned to love zone defense*. *The Guardian*. Retrieved 11 December 2016, from https://www.theguardian.com/sport/2015/nov/25/nba-learned-love-zone-defense

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., & Grisel, O. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal Of Machine Learning Research*, *12*, 2825-2830. Retrieved from http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

Puranmalka, K. (2013). *Modelling the NBA to Make Better Predictions* (Master's Thesis). Retrieved from https://dspace.mit.edu/handle/1721.1/85464

Sokolova, M. & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427-437. http://dx.doi.org/10.1016/j.ipm.2009.03.002

Stats LLC. (2015). *Introduction to the STATS API*. Retrieved from http://www.stats.com/webinars/introduction-to-the-stats-api/

Stats LLC. (2016). *Basketball Data Feed | Basketball Player Tracking | SportVU*. *STATS.com*. Retrieved 21 September 2016, from http://www.stats.com/sportvu/sportvu-basketball-media/

Wang, K. & Zemel, R. (2016). Classifying NBA Offensive Plays Using Neural Networks. In *10th MIT Sloan Sports Analytics Conference*. Boston. Retrieved

from http://www.sloansportsconference.com/wp-content/uploads/2016/02/1536-Classifying-NBA-Offensive-Plays-Using-Neural-Networks.pdf

Wiens, J., McQueen, A., & Guttag, J. (2014). Automatically Recognizing On-Ball Screens. In *8th Annual MIT Sloan Sports Conference*. Boston. Retrieved from http://www.sloansportsconference.com/wp-content/uploads/2014/02/2014_SSAC_Recognizing-on-Ball-Screens.pdf

Witten, I., Frank, E., Hall, M., & Pal, C. (2017). *Data mining* (1st ed.). Amsterdam: Elsevier.

Zillgitt, J. (2012). *The zone defense has found its place in the NBA. USATODAY.COM*. Retrieved 11 December 2016, from http://usatoday30.usatoday.com/sports/basketball/nba/story/2012-01-17/zone-defense-has-found-its-place-in-the-nba/52657598/1