5-2021

Machine Learning Approaches to Dribble Hand-off Action Classification with SportVU NBA Player Coordinate Data

Dembe Stephanos
East Tennessee State University

Machine Learning Approaches to Dribble Hand-off Action Classification with SportVU NBA Player Coordinate Data

---

A thesis

presented to

the faculty of the Department of Computing

East Tennessee State University

In partial fulfillment

of the requirements for the degree

Master of Science in Computer Science, Applied Computer Science

---

by

Dembé Koi Stephanos

May 2021

Dr. Ghaith Husari, Chair

Dr. Brian T. Bennett

Mr. Mathew Harrison

Dr. Ferdaus Kawsar

Keywords: machine learning, classification, deep learning, basketball analytics, feature engineering, hexbin mapping

ABSTRACT

Machine Learning Approaches to Dribble Hand-off Action Classification with SportVU NBA Player Coordinate Data

by

Dembé Koi Stephanos

Recently, strategies of National Basketball Association teams have evolved with the skillsets of players and the emergence of advanced analytics. One of the most effective actions in dynamic offensive strategies in basketball is the dribble hand-off (DHO). This thesis proposes an architecture for a classification pipeline for detecting DHOs in an accurate and automated manner. This pipeline consists of a combination of player tracking data and event labels, a rule set to identify candidate actions, manually reviewing game recordings to label the candidates, and embedding player trajectories into hexbin cell paths before passing the completed training set to the classification models. This resulting training set is examined using the information gain from extracted and engineered features and the effectiveness of various machine learning algorithms. Finally, we provide a comprehensive accuracy evaluation of the classification models to compare various machine learning algorithms and highlight their subtle differences in this problem domain.

DEDICATION

Dedicated to my brilliant, wonderful wife, Emma. Thanks for listening to me rant on about basketball analytics since way before it was considered graduate work. I love you.

ACKNOWLEDGEMENTS

Thank you to my committee members, Dr. Bennett and Prof. Harrison, for their guidance and support. Additional thanks to Jacob Hoyos and Dave Schmidt for their assistance and encouragement, and to Neil Seward for his contributions to getting me started. Most of all, to Dr. Husari, thank you for your excellent mentorship, without which, this thesis would not exist.

TABLE OF CONTENTS

ABSTRACT 2
DEDICATION 4
ACKNOWLEDGEMENTS 5
LIST OF FIGURES 9
Chapter 1. Introduction 11
1.1 Basketball Strategy 11
1.2 Key Terms 12
1.3 Dribble-Hand-Off 13
1.3.1 Fake Variants 14
1.4 Remaining Organization of Thesis 16
Chapter 2. Background 17
2.1 NBA Player Tracking Data 17
2.1.1 SportVU 17
2.1.2 Event Labels 18
2.2 Related Works 20
2.2.1 Supervised Learning 21
2.2.2 Unsupervised Learning 24
2.3 Classification Models 27
2.3.1 Support Vector Machine 28
2.3.2 Decision Tree 29
2.3.3 Gaussian Naïve Bayes 31
2.3.4 Multilayer Perceptron Neural Network 32
2.4 Challenges and Opportunities 33
2.4.1 Feature Engineering and Extraction 34
2.4.2 Deeper Information Discovery of Actions 34
2.4.3 Effective Video Preprocessing and Noise Filtration 35
2.4.4 Annotation Scalability 36
Chapter 3. Design and Implementation 37
3.1 Approach Design and Architecture 37
3.1.1 Preprocessing 38
3.1.2 Candidate Algorithm 39

3.1.3 Manual (Human) Labeling 40
3.1.4 Hexbin Mapping 40
3.1.5 Feature Generating 41
3.1.6 Model Training 41
3.1.7 Model Testing 42
3.2 Candidate Algorithm 43
3.2.1 Identifying Passes 44
3.2.2 Selecting Candidates 44
3.2.3 Evaluating Selected Candidates 45
3.3 Hexbin Positional Conversion 46
3.4 Feature Creation 47
3.4.1 Extracted Features 48
3.4.2 Engineered Features 49
3.5 Entity Overview 51
Chapter 4. Evaluation 53
4.1 Accuracy Measures 53
4.1.1 Metric Collection Methodology 54
4.2 Evaluating Features 55
4.2.1 Information Gain 55
4.2.2 Impact of Hexbin Mapping on Model Performance 57
4.3 Comparing Classification Models 57
4.3.1 Support Vector Machine 59
4.3.2 Decision Tree 60
4.3.3 Gaussian Naïve Bayes 61
4.4 Comparing Deep Learning Models 62
Chapter 5. Conclusion 66
5.1 Discussion of Learning Models 67
5.2 Cross-domain Application of this Approach 67
5.3 Future Work 68
References 69
VITA 72

LIST OF TABLES

Table 1. An Overview and Description of the Various Event Labels in the Dataset 19

Table 2. Resulting Play-groups Using Cluster Analysis..... 26

Table 3. A Summary of the Total Entities Present Within the Training Set 45

Table 4. Overview of Extracted Features (42 total)..... 49

Table 5. Overview of Engineered Features (16 total)..... 51

Table 6. Confusion Matrices for Classic Classification Models 59

Table 7. Confusion Matrices for Deep Learning Models 65

LIST OF FIGURES

Figure 1. An Illustration of an On-ball Screen Broken into Approach and Execution Stages 13

Figure 2. An Illustration of a Dribble-hand-off Broken into Approach and Execution Stages 14

Figure 3. An Illustration of a Fake Dribble Hand-off 15

Figure 4. Overview of SportVU JSON Containing Data for a Single Game Capture 18

Figure 5. Taxonomy of Machine Learning Approaches Applied in the Domain 20

Figure 6. K-means Clustering Algorithm 25

Figure 7. Principal Component Analysis Algorithm 27

Figure 8. Illustration of the SVM Optimal Hyperplane with a Linear Dataset..... 29

Figure 9. Simplified Decision Tree for the DHO Candidate Algorithm..... 30

Figure 10. Classifying Sample Using Naïve Bayes and Gaussian Distributions..... 31

Figure 11. A Depiction of a Three-layer Dual-input/Dual-output Artificial Neural Network 32

Figure 12. Finding the Ball-handler Algorithm 35

Figure 13. The Dribble-hand-off Classification Pipeline 38

Figure 14. Steps in the candidate algorithm selection process 43

Figure 15. Hexbin Mapping of Coordinate Data for DHO Action 47

Figure 16. Entity Diagram of the Classification Pipeline Domain 52

Figure 17. Overview of Information Gain per Feature 56

Figure 18. Comparing Accuracy Metrics Across Classic Classification Models 58

Figure 19. Learning and ROC Curves for SVM Model..... 60

Figure 20. Learning and ROC Curves for DT Model 61

Figure 21. Learning and ROC Curves for GNB Model..... 61

Figure 22. Comparing Accuracy Metrics Across Deep Learning Models..... 62

Figure 23. Comparing ROC Curves of Neural Networks..... 63

Figure 24. Learning Curve for Scikit-learn MLPClassifier 63

Figure 25. Learning Curve for Keras Neural Network 64

Chapter 1. Introduction

A recent application of machine learning is the study of strategies and patterns in major sports. Since 2012, when SportVU first started capturing player location data during National Basketball Association (NBA) games, there have been myriad pursuits to parse this raw data and extract meaningful features about players and the game itself. Professional basketball is a fast-paced, complex system that is continuously evolving, making high-level features challenging to obtain. As a result, most work identifying and analyzing individual plays or actions within a game is still performed by humans watching game tape. Several insightful approaches have been attempted to remedy this using modern machine learning tools like support vector machine (SVM) classifiers, neural networks, and clustering algorithms to automate these time-consuming tasks [1]–[6].

Many of these approaches seek to limit the problem domain by focusing on a particular play or aspect of the game, but a coherent end-to-end system that parses raw data and provides AI-driven recommendations has yet to be fully realized. This thesis will examine and critique some of the more effective approaches to pattern mining NBA player movement data, focusing primarily on the most researched play action in literature: the pick-and-roll. This thesis then proposes and evaluates an architecture for a classification pipeline for detecting a different action: the dribble hand-off.

1.1 Basketball Strategy

Basketball is a team sport in which two teams of five active players compete in timed possessions to amass as many points as possible by scoring, or getting the basketball into their goal, a circular rim positioned ten feet in the air with a glass backboard. Several restrictions are placed on players, who must dribble (bounce) the ball when moving and may not leave the

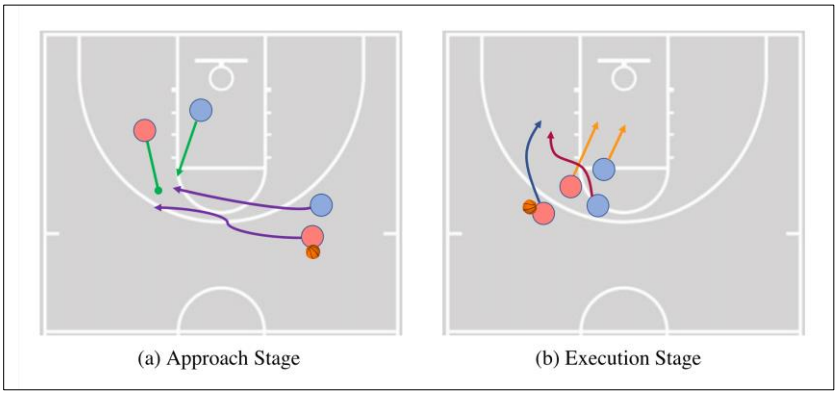
bounds of the rectangular court. In the NBA, games involve as many as fifteen players per team, with twenty-four second possessions spanning over four, twelve-minute quarters. Traditionally, a variety of plays are performed in sequence with the objective of creating valuable scoring opportunities. Recently, these strategies have significantly evolved from long-scripted plays to read-and-react offenses, in which a variety of independent actions are performed by players who then read the defensive response and react accordingly.

1.2 Key Terms

First, there are a few key terms for that must be defined. Perhaps most importantly are two already used in this thesis: play and action. For the purposes of this thesis, a play is defined as a strategic and intentional sequence of actions taken out by cooperating offensive players in an effort to create the space required for a valuable scoring attempt. An action is defined as a discrete interaction between two or more offensive players and their corresponding defenders. Since only one player may be in possession of the ball at a given time, many of these plays and actions involve what is called a screen, in which an offensive player positions themselves firmly between the ball-handler and a defending player, forcing that defending player to navigate around them, subsequently creating space for the non-screening offensive player.

There are many versions of screens, each with several additional variants, but perhaps the most common is the on-ball screen. An on-ball screen occurs when an offensive player sets a screen for the ball-handler, and the ball-handler then guides their defender into the screen. This can be broken in to two stages: the approach and the execution, Figure 1 illustrates the approach and execution stages [6].

Figure 1. *An Illustration of an On-ball Screen Broken into Approach and Execution Stages*

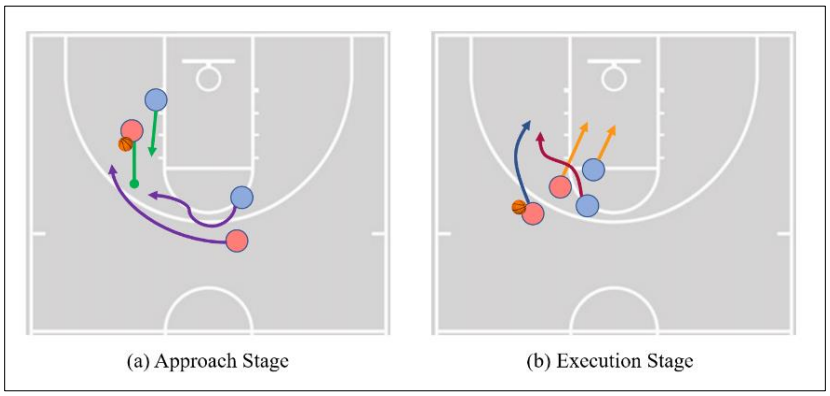


The on-ball screen is prominent in many plays and can even be considered a play itself, often referred to by one of its most common variants, the pick-n-roll. The on-ball screen is a favorite in the machine learning literature [1], [2], [4], [7], [8] as it is a frequently-occurring action with an easily identifiable set up and great variety of possible executions.

1.3 Dribble-Hand-Off

To further explore the potential applications of machine learning to pattern extraction and classification in NBA player movements, this thesis targets a less explored action within the literature: the dribble hand-off (DHO). Much of the prior research into action classification has been focused on the on-ball screen, which similarly to the DHO requires two coordinating offensive players in screening and cutting roles, involves an on-ball action often implemented in more complex plays, and can be performed in a number of variants in which either player may attempt to score. As shown in Figure 2, a dribble-hand off occurs when the screening player (often a taller frontcourt player such as a Center or Power Forward) dribbles into a screen and the cutting player (often a smaller player like a Point Guard or Shooting guard) runs to meet them. Once the cutting player gets close enough, the screening player “drops-off” the pass and then seeks to become an obstacle for the opponent defending the cutting player. In this example, the screener and cutter and their corresponding defenders all then roll (run) towards the basket.

Figure 2. *An Illustration of a Dribble-hand-off Broken into Approach and Execution Stages*

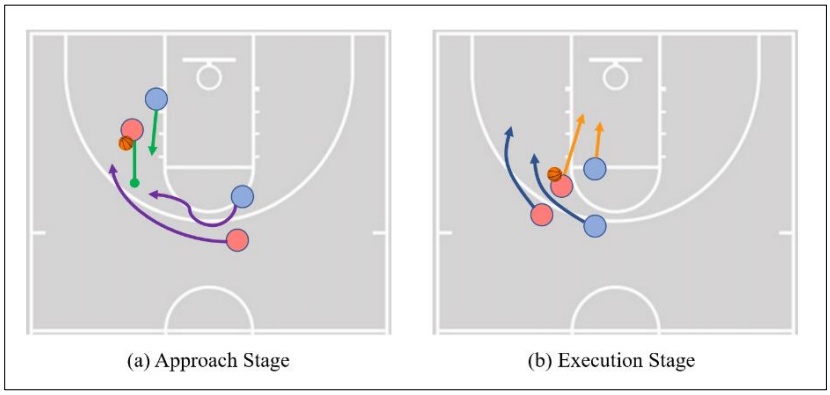


It is interesting to note here that the execution stage in this DHO example is the same as that of the on-ball screen. While both can occur in a significant number of variants in both the approach and execution, the similarities they share in being on-ball actions means they are often implemented in similar situations with similar desired results.

1.3.1 Fake Variants

Of the many DHO variants, there is one set that requires some additional explanation: fakes. In basketball and other sport strategy, fake plays or actions are often implemented by teams alongside more traditional variants as a way of confusing the defense and creating opportunity. A fake DHO has a very similar or even identical approach stage to other DHO strategies, but instead of completing the pass, the screening player will allow the cutter to run under the ball before setting up for the next action or driving to the basket themselves. Figure 3 contains an illustration of a fake DHO that occurs with the same approach stage as the previous example in Figure 2.

Figure 3. *An Illustration of a Fake Dribble Hand-off*



The existence of fakes presents a unique challenge in classifying DHOs. When performing an on-ball screen, there are variants where the ball-handling player 'denies' the screen and instead drives in the opposite direction or the screening player 'slips' on the screen and instead of establishing position, rolls toward the basket. Both variants could be considered fake plays because the typical execution is being actively diverted in an intentional effort to confuse the defending players. The difference here is that the approach and execution stages are still very similar, and the key distinguishing feature of a screen being set for the ball-handler is satisfied. In the case of the fake dribble hand-off, the hand-off itself does not occur. This can make identification and classification more difficult because the intention of some actions is to divert from the conventional patterns.

Within this thesis, fake dribble hand-off actions are labeled and grouped with conventional DHOs but given an additional note tag identifying it as a fake. While including these examples likely imposes a lower ceiling on the classification capabilities of the pipeline, overcoming the diversity in strategic variance present in the dataset is a known challenge of this thesis and removing those examples would require arguing they are not dribble hand-offs, which cannot be justified. Ultimately, future work to create a separate classifier to identify particular variants, such as fakes, may be necessary.

1.4 Remaining Organization of Thesis

The remainder of this thesis is organized as follows. In section 2, there is a discussion of the dataset and the technology used to track and collect positioning data of the ball and players during a game as well as a systematic and detailed review of state-of-the-art studies on the different types of machine learning systems applied and proposed in this domain. In section 3, an overview of the pipeline construction and architecture is presented, and in section 4 the results of that pipeline are evaluated by various machine learning models. A conclusion is drawn and avenues for potential cross-domain applications and future work are offered in section 5.

Chapter 2. Background

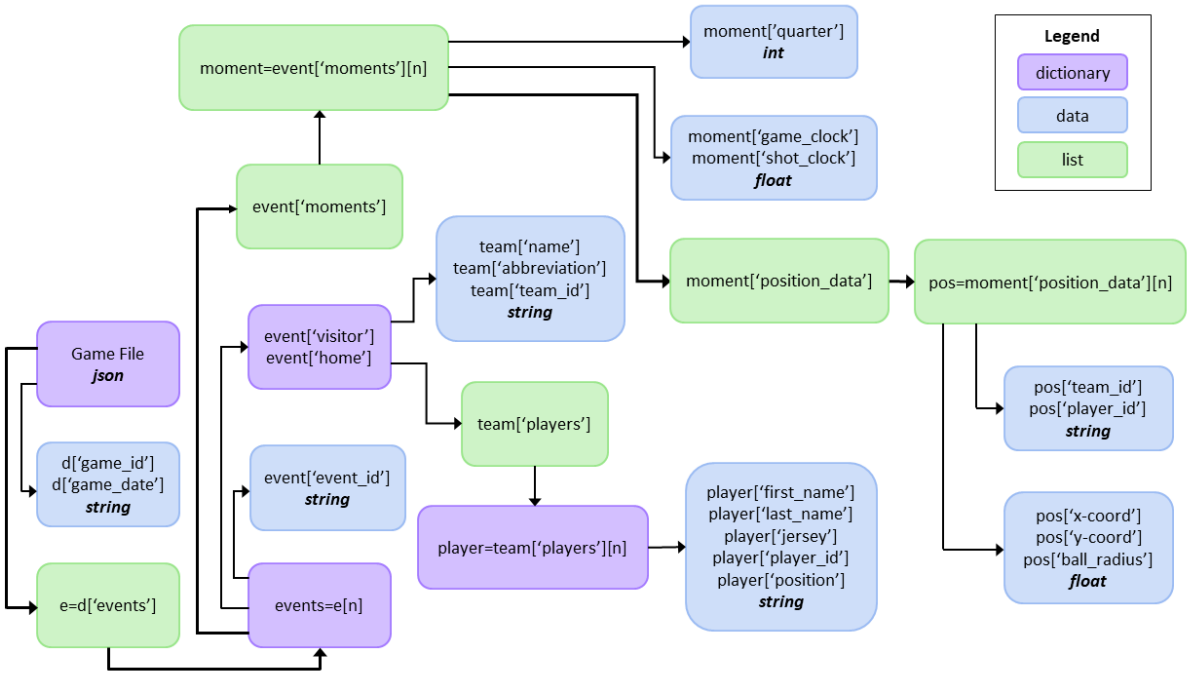
2.1 NBA Player Tracking Data

The pipeline depends on two separate data sources, both indexed by a common event id. The first of these datasets contains the player tracking data for a given game, while the second provides event descriptions and additional information for events, such as the score, the type of shot taken, the players involved in the action, and the time and quarter it occurred in. These datasets are intended to overlap, but in practice, there are often synchronization issues resulting in incorrect event label assignments to coordinate data. This thesis focused on 15 games for which these inconsistencies were limited.

2.1.1 SportVU

In 2012, the NBA began tracking player movements during games by installing a total of six cameras that capture the location of all ten players on the court, as well as the ball and three referees, twenty-five times per second. This data was publicly available through the 2015-2016 season and inspired a great deal of interest among the sport's analytic community, eventually making its way to machine learning experts [9]. The data itself is collected in a JSON object, typically around 600 MB in size, containing a variety of defining information such as the teams and players involved, the date and location of the game and the coordinates of the players broken up by quarter and event. Figure 4 illustrates a detailed breakdown of the JSON object [6].

Figure 4. *Overview of SportVU JSON Containing Data for a Single Game Capture*



An event occurs when one team takes possession of the ball and ends either when they score, a foul or turnover is committed or in other unique circumstances, such as clock malfunction. Since the raw positional data of the players does not paint a full picture, this dataset is often combined with other event-specific data, such as the type of shot taken or other play-specific details that allow for a more cohesive view of on-court behaviors. Many techniques are used to abstract this low-level data, and most of these include the embedding of the coordinates into vectors or trajectory images that represent a player's full range of motion over the course of a given possession.

2.1.2 Event Labels

To provide additional context and higher-level descriptions to the raw coordinate data, event labels are loaded and associated with the SportVU events by their corresponding event

numbers. These event labels were originally created by researchers in (Saini & Evans, 2017) and a summary of them can be found in Table 1.

Table 1. *An Overview and Description of the Various Event Labels in the Dataset*

Column Name	Description
EVENTMSGACTIONTYPE	type of action (enum)
EVENTMSGTYPE	further breakdown of action (enum)
EVENTNUM	event number/order in game
GAME_ID	NBA game id
HOMEDESCRIPTION	raw textual description of event for home team action
NEUTRALDESCRIPTION	raw textual description of event if neutral action
VISITORDESCRIPTION	raw textual description of event for away team action
PERIOD	current period
SCORE	game score
SCOREMARGIN	difference of game score (home - away)
WCTIMESTRING	real world time
PCTIMESTRING	time left in quarter
PLAYER1_ID	NBA player id for player 1 (in play-by-play description)
PLAYER1_NAME	player 1 name
PLAYER1_TEAM_ABBREVIATION	team abbreviation for player 1
PLAYER1_TEAM_CITY	team city for player 1
PLAYER1_TEAM_ID	team id for player 1
PLAYER1_TEAM_NICKNAME	team nickname for player 1
PLAYER2_ID	NBA player id for player 2 (in play-by-play description)
PLAYER2_NAME	player 2 name
PLAYER2_TEAM_ABBREVIATION	team abbreviation for player 2
PLAYER2_TEAM_CITY	team city for player 2
PLAYER2_TEAM_ID	team id for player 2
PLAYER2_TEAM_NICKNAME	team nickname for player 2
PLAYER3_ID	NBA player id for player 3 (in play-by-play description)
PLAYER3_NAME	player 3 name
PLAYER3_TEAM_ABBREVIATION	team abbreviation for player 3
PLAYER3_TEAM_CITY	team city for player 3
PLAYER3_TEAM_ID	team id for player 3
PLAYER3_TEAM_NICKNAME	team nickname for player 3

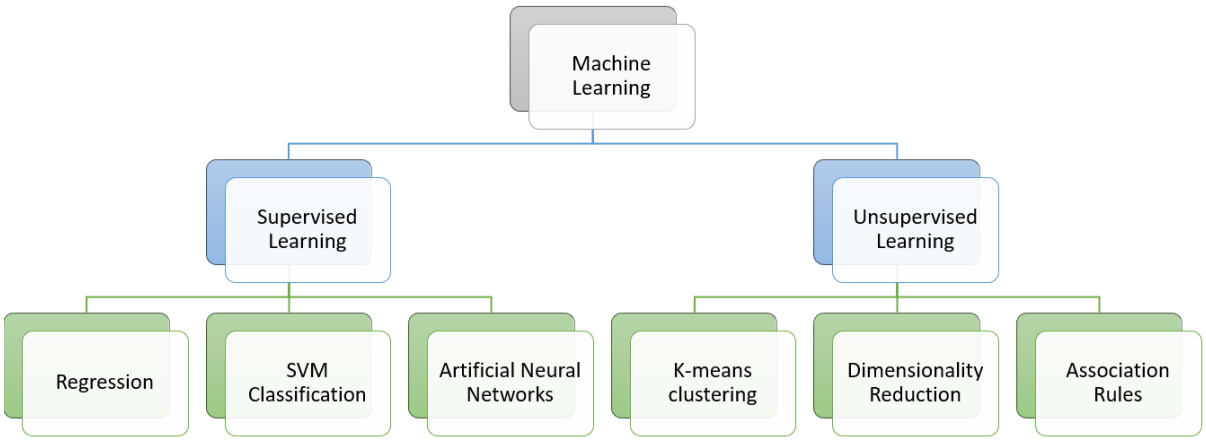
These event labels not only associate game actions with the players involved and the resulting scores, they also contain high-level action type designations. These designations can be used to filter out the many events that do not contain data of interest, for example, at the

beginning and end of quarters or when a player substitutes into the game. There were some inconsistencies between the available event labels and SportVU datasets, which requires note. This resulted in incorrect and inconsistent event labeling, which led to unexpected and undesirable results. While, initially, some efforts were taken in this thesis to correct these issues, there were enough games without problematic synchronization to build the training and testing sets. It was ultimately deemed out of the scope of this thesis to address this issue further.

2.2 Related Works

In this section are two broad categories to review significant research published on machine learning-based NBA analytics: supervised (classification) approaches and unsupervised (including clustering) approaches. A broad taxonomy of these approaches is found in Figure 5.

Figure 5. *Taxonomy of Machine Learning Approaches Applied in the Domain*



In the case of supervised machine learning, a human researcher chooses an event (e.g., an action) that occurs in the data, and then a classification model is built to automatically find all occurrences of that event in the dataset. Unsupervised machine learning, on the other hand, examines the naturally occurring patterns within the data, and analyzes these patterns and their

frequencies to potentially reveal new knowledge, groupings, or strategies in the data. While these approaches are different, they are often used together to identify actions and examine general patterns. The key points of each publication are discussed in the following subsections.

2.2.1 Supervised Learning

Supervised learning infers a function from labeled training data to create knowledge structures that support the process of classifying new instances into a set of predefined classes [11]. The input for this type of learning algorithms is a collection of sample instances that are pre-classified (labeled) into a predefined set of classes. The output of this process is a classification model that is constructed by analyzing the training data and producing an inferred function that can determine the class (label) for unseen instances with a reasonable accuracy. There are two key steps in supervised learning:

- **Training**: Analyzes the training data (labeled instances) and constructs a classification model
- **Testing**: Uses the constructed (trained) model to classify new instances and reports the accuracy

There are various supervised learning algorithms that differ in their approaches to analyzing, inferring, and generalizing knowledge from the labeled training data to construct the classification model.

Due to their top-down nature, supervised learning efforts in NBA analytics are often focused on identifying and labeling certain events in the dataset. In the case of the on-ball screen, research works formulate and utilize a set of rules configured around the distances and durations of time that coordinating offensive players and the ball spend in proximity to each

other. Approaches that use a sliding window to extract features of a certain event are frequently employed to ensure the capturing of all actions, as it is very difficult to specify the point in a given possession at which an action has started [12]. Using a sliding window, the screen moment can be specified by identifying the moment right before the players begin moving again [2].

Yu and Chung [6] propose an SVM binary classifier to automatically identify on-ball screens. First, they propose an algorithm to identify event candidates that may contain on-ball screens. This algorithm uses a set of rules that are devised manually to extract on-ball screen candidates and filter out other events. These candidates are then analyzed manually (by humans) to determine and label each as on-ball screen or otherwise. Once the labeled dataset (actions and their labels) is created, the set is split into two parts: a training set and a testing set. This split is often a 9:1, where 90% of the data is used for training and 10% is used for testing. The SVM binary classifier was trained using four distance- and speed-based player features. After the training phase, the testing set was fed to the classifier to evaluate its performance in identifying on-ball screens, and it achieved a recall of 90.46%.

McQueen and Guttag [2] propose an approach that utilizes a set of rules to identify actions. The data are segmented into periods of time around these actions. Using this approach, they extracted 30 continuous features from each action. Once extracted, these features were discretized into five binary features based on quintiles. This resulted in a 150-dimensional feature vector for every action. The data were then labeled and split into approximately 51.9% training and 48.1% validation sets. Then, an SVM linear classifier was constructed using the training data. This process was repeated 3,200 times using different splits of data. The classifier achieved a recall of 82% and precision of 80%.

McIntyre et al. [1] use a more robust dataset consisting of 270,823 ball screens. The labeled dataset was split into 70% training and 30% testing sets. Four classifiers were trained to identify four different types of ball screens, achieving a recall of 83% and precision of 78% in identifying the type of on-ball screens where the defender stays between the ball handler and the screener.

An approach proposed by Kates [12] uses the SportVU data to detect six different plays, where each play has 20 labeled instances at minimum. The approach trained six SVM classifiers (one for each play type) using 8:2 stratified data split. The classifiers predicted the plays with a collective accuracy of 72.6% and an F-score of 72.7%.

Artificial Neural Network (ANN) is a learning algorithm that is based on a set of connected nodes. Each connection between two nodes communicates a signal to the destination node. This signal is a real number and the output of each node is calculated by a nonlinear function that aggregates the node's inputs in a process that is loosely based on the biological neural networks in the human brain. Wang and Zemel [5] use variants of neural networks to automatically classify play sequences into eleven selected play classes (or offense strategies). Recurrent Neural Network (RNN) achieved the highest accuracy, with a top-three accuracy of 80%, in classifying ninety-five unlabeled sequences (approximately 6% of the data) into eleven possible play classes. As expected, the simple neural networks achieved a lower top-three accuracy (77%) when compared with RNN. This is due to the ability of RNN to better handle and learn from sequential data of variable length as it accumulates change over time.

A current limitation of this practice is that researchers are responsible for creating the labeled dataset. While this is likely okay for more explicit play-actions like the on-ball screen, it introduces a level of bias, is less suited for more subtle or nuanced actions, and would be better

performed by a domain expert [6] These supervised learning approaches work well for the base tasks of identification and classification and are thus an excellent first step toward a more robust pipeline capable of providing real analytic insights. Despite the high accuracy, an on-ball screen dataset alone cannot provide much insight into the problem domain. This is where unsupervised approaches can be more effective and provide more insightful analysis. These approaches can take the refined data produced by the supervised approach as inputs and produce interesting, novel insights into the patterns and strategies present in the player movement, narrowing those patterns to specific variations in both the approach and execution of the on-ball screens identified by the supervised approach.

2.2.2 Unsupervised Learning

Unsupervised learning (or clustering) is a type of machine learning that allows a model to work with minimal or no human supervision by finding the natural clusters (groups) in the data using heuristics without reference to outcomes labeled by humans [13].

Clustering focuses on finding patterns in the data and then creates groups of instances with similar properties. This similarity is calculated by a distance function, such as Euclidean distance. Depending on the approach, these clusters can be exclusive, in which instances belong to only one cluster, or they can be overlapping, where an instance could belong to more than one cluster. Another type of clustering is probabilistic clustering, where there is a certain probability that an instance belongs to a cluster. Hierarchical clusters occur when there are parent clusters at the top level, and each of these parent clusters is further refined to smaller, more specific clusters [14]. There are three types of clustering methods:

- **K-means algorithm**: Forms clusters in numeric domains, grouping instances into exclusive (disjoint) clusters. Figure 6 shows the pseudocode of K-means clustering [15].

- **Incremental clustering**: Creates a hierarchical grouping of instances.
- **Probability-based methods**: Assign an instance to a cluster based on a probability score for membership.

Figure 6. *K-means Clustering Algorithm*

Algorithm 1: K-means Clustering

Input: K, and unlabeled data set $\{x_1, \ldots, x_N\}$.
Output: Cluster centers $\{s_k\}$ and the assignment of the data points $\{\pi_{nk}\}$
Randomly initialize $\{s_k\}$
repeat
  for $n := 1$ *to* N **do**
    for $k := 1$ *to* K **do**
      if $k = argmin_i ||s_i - x_i||^2$ **then**
        $\pi_{nk} := 1$
      else
        $\pi_{nk} := 0$
      end
    end
  end
  for $k := 1$ *to* K **do**
    $s_k := \frac{\sum_n x_n \pi_{nk}}{\sum_n \pi_{nk}}$
  end
until $\{\pi_{nk}\}$ *or* $\{s_k\}$ *don't change;;*

Unsupervised learning is becoming increasingly popular among sports scientists. Research works that utilize unsupervised learning methods are geared more towards pattern extraction and strategy analysis than supervised efforts are.

Nistala and Guttag [3] and Nistala [16] utilize cluster analysis to assign similar attacking movements into groups, such as movements along sidelines, run along the baseline, and other attack movements. First, they constructed 3 million trajectory images from NBA player tracking data collected by the STATS SportVU player tracking system [9] They then utilized a numerical abstraction algorithm for player movements and ran the K-means clustering algorithm on the 3 million trajectory images to group similar trajectories together. The algorithm grouped the

attacking player movements into 20 clusters. Manual evaluation of these clusters (by selecting case studies from each one) showed that deep learning can be used to interpret patterns of basketball attack movements.

Brooks [7] proposes an approach utilizing unsupervised machine learning to characterize patterns of play for NBA teams on offense. First, the approach builds an image for each player's movement on offense to give a starting point for comparing player movements across different possessions. They then utilized the k-means clustering algorithm using the Python package scikit-learn [17] to cluster 60,000 instances of possessions. The number of clusters was set to $k = 30$, as it represented the "knee" in the response curve between k and the average within-cluster distance. This process produced thirty clusters each containing around 2,000 instances. The thirty resulting clusters were evaluated manually (using human judgement) by checking the top ten instances that are closest to the center of the cluster. This evaluation confirmed that instances in the same cluster had similar patterns of movement for the players and the ball. Each cluster was then given a description that describes the movement pattern (or play) of its instances.

Table 2 shows a sample of five clusters with their descriptions [7].

Table 2. *Resulting Play-groups Using Cluster Analysis*

Cluster	Description
1	*On-ball screen, pass screener or far side corner*
2	*Screen to the right of the top of the key, pass to the right corner*
3	*On-ball screen on right elbow, drive or pass to the center*
4	*Left side post-up*
5	*Pick-and-roll at the top of the key*

Lutz et al. [18] propose an approach that uses statistics such as field goals and assists to cluster similar players into 10 categories. Franks et al. [19] utilize non-negative matrix factorization (NMF) to cluster defensive players based on the locations of field goals.

Sampaio et al. [20] cluster players with similar features, such as attacking, defense, and passing statistics. Sampaio et al. [21] and Teramoto et al. [22] apply dimensionality reduction techniques, such as principal components analysis (PCA), in basketball. Figure 7 shows a pseudo code for PCA [15].

Figure 7. *Principal Component Analysis Algorithm*

Algorithm 3: Principle Component Analysis

Input: L, and input vectors of an unlabeled or labeled data set $\{x_1, \ldots, x_N\}$.

Output: The projected data set $\{x_1, \ldots, x_N\}$, and basis vectors $\{w_j\}$, which form the principal subspace

$\bar{x} := \frac{1}{N} \sum_n x_n$

$S := \frac{1}{N} \sum_n (x_n - \bar{x})(x_n - \bar{x})^T$

$\{w_j\}$:= the L eigenvectors of S corresponding to the L largest eigenvalues

for $n := 1$ *to* N **do**

  for $j := 1$ *to* L **do**

    $z_{nj} := (x_n - \bar{x})^T w_j$

  end

end

2.3 Classification Models

In this thesis, several machine learning models were constructed and fit to the training set to evaluate its effectiveness. These machine learning algorithms were selected as a sample to represent a variety of other similar approaches and can be broken into two groups: supervised classification models and deep learning neural networks. This thesis evaluated the Scikit-learn implementations of support vector machine, decision tree, Gaussian Naïve Bayes, and multi-

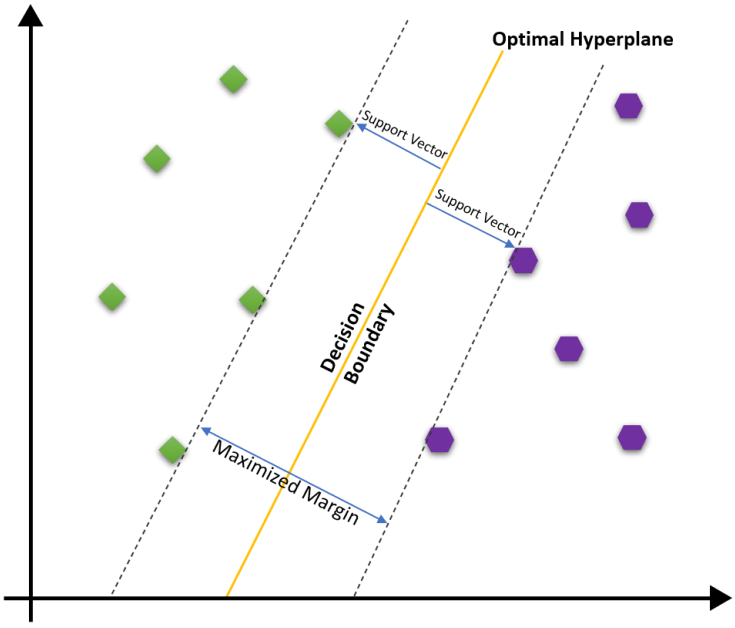
layer perceptron classifiers as well as a Keras/TensorFlow multi-layer neural network. Additional descriptions and explanations of each classifier can be found below.

2.3.1 Support Vector Machine

Support vector machine (SVM) is one of the more powerful and frequently-implemented classification models in the machine learning literature and a favorite of other NBA action classification pipelines. One such pipeline using SVM to classify on-ball screens was optimized to reach a sensitivity of 95% for on-ball screens [6]. This thesis uses the Scikit-learn SVC model, which is a widely used implementation of the SVM algorithm.

SVM is a linear model capable of tackling classification and regression problems and functions by calculating a hyperplane to separate a dataset. Other linear classifiers operate in a similar manner but do not account for how close their estimations are to the data instances and, as a result, often perform poorly on unseen data. SVM seeks to remedy this problem by choosing a separating line that maximizes the margin on either side of the predictive line, resulting in the optimal hyperplane [23]. An illustration of an SVM hyperplane is provided in Figure 8 [24]. As datasets grow more complicated and feature sets grow, SVM implements more involved kernel functions to evaluate the similarity of samples and construct more fine-tuned predictive curves.

Figure 8. *Illustration of the SVM Optimal Hyperplane with a Linear Dataset*

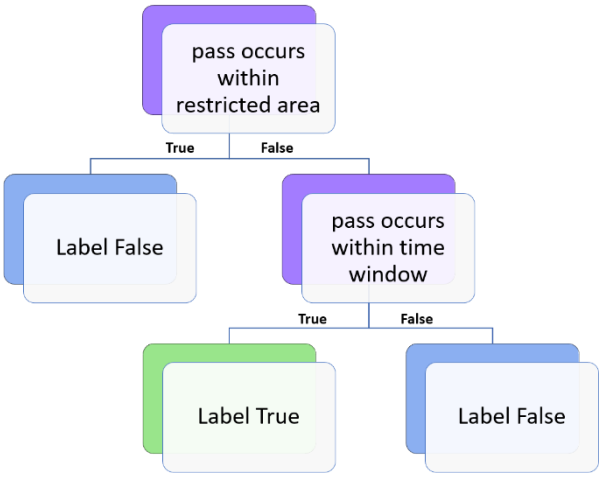


Considering its prevalence in the literature, SVM is a clear choice for evaluating the predictive capabilities of the thesis dataset. Its similarity to other linear classifiers, such as linear regression, also provides insight into how similar algorithms might perform.

2.3.2 Decision Tree

A decision tree (DT) is another powerful machine learning algorithm that, in the vein of SVM, is capable of both classification and regression [25]. They can be combined to form more complex and powerful random forests, but the core concept is simple enough to visualize. Figure 9 contains a basic decision tree representing the candidate algorithm.

Figure 9. *Simplified Decision Tree for the DHO Candidate Algorithm*



At its core, a DT contains a tree-like model of choices and consequences that filter down to conclusive nodes. In the example above, there are two nodes that represent tests on the data, colored in purple. The tree is traversed by passing or failing each test and proceeding in the corresponding direction down the tree. In this simplistic case, there are at most two decision nodes to visit and only one valid path that results in a positive labeling. This thesis uses the Scikit-learn decision tree classifier and is fitted on the generated feature vectors, resulting in a far more complex tree.

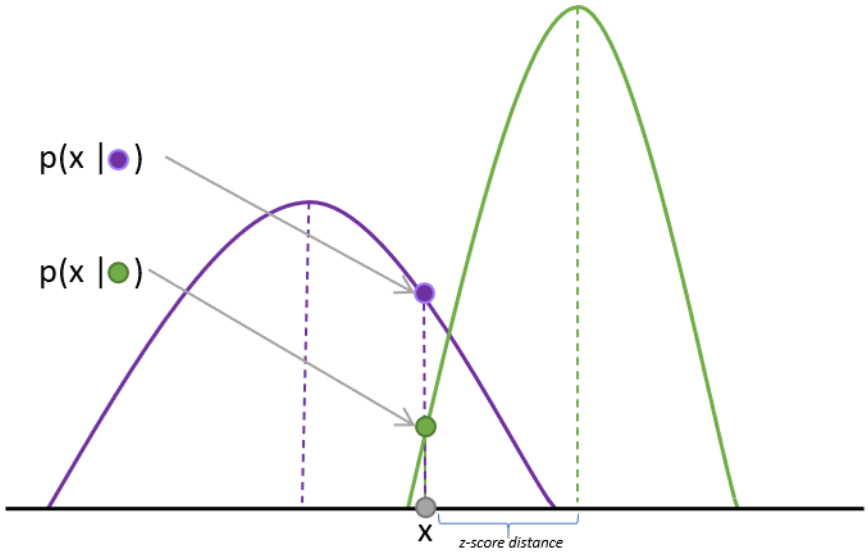
In these more complicated cases, trees can grow very large and contain a significant number of nodes, but, due to their generally being approximately balanced, traversing the tree for a prediction only requires $O(log_2(m))$ steps, where m is the height of the tree [26]. This is because many decision tree algorithms construct the tree by finding splits for the data that will reduce impurity, recursively adding splits until no more suitable options are found, resulting in a more or less binary tree structure. Decision trees are prone to overfitting. As a result, they require regularization to produce consistent models. Decision tree was chosen as a classifier for this

thesis because of its prevalence in the literature and as a representative of other tree-based learning models.

2.3.3 Gaussian Naïve Bayes

Naïve Bayes classification models are a family of probabilistic classifiers whose core unifying feature is that they treat all values of the feature vector independently, assuming zero correlation between them. Gaussian builds upon this by further assuming that the instances of the various potential classifications are distributed normally. It then takes each value from the feature vector and compares the *z-score* distance between that point and the class-mean for the different potential classifications [27]. Evaluating the overall proximity of each sample to the class means provides the probabilistic class designation. An example of this can be found in Figure 10 [28]:

Figure 10. *Classifying Sample Using Naïve Bayes and Gaussian Distributions*

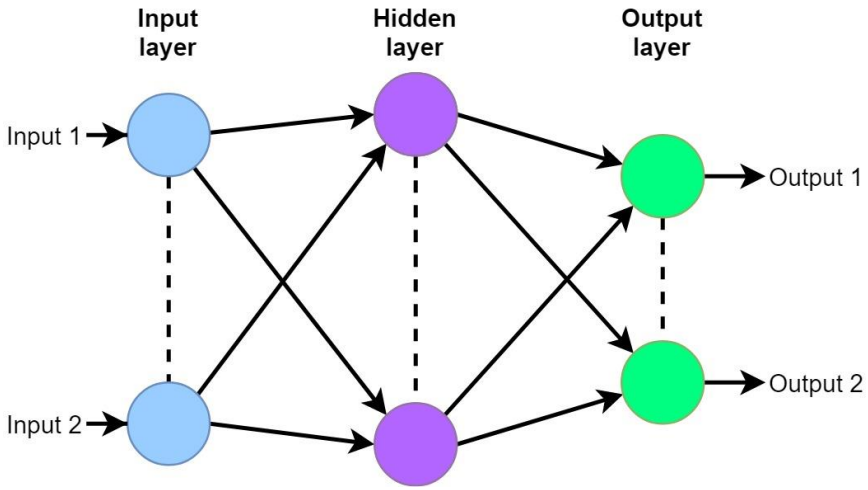


Here, two classes (purple and green) both have their normalized distributions plotted. The sample *X* has its *z-score* distance calculated from the average of the class distributions and is found to be more similar to class green and will be classified as such.

2.3.4 Multilayer Perceptron Neural Network

A multilayer perceptron (MLP) is a feedforward artificial neural network that uses backpropagation learning and is adept at classification, recognition, and prediction. It consists of three layer types: input, output, and hidden [26]. Each layer of the neural network is fully connected to the perceptrons (nodes) in the following layer. An illustration of these layers can be found in Figure 11 [29].

Figure 11. *A Depiction of a Three-layer Dual-input/Dual-output Artificial Neural Network*



Here, two inputs representing different descriptive features are fed into the input layer. Each perceptron in the neural network uses a nonlinear activation function and independently assesses the features it receives from the previous layer to determine its own weighting system to describe how the feature data should be interpreted for optimal classification. The output of each node from a non-input layer of perceptrons is then combined with a bias node whose input is 1 (to ensure a constant is passed to the next layer), at which point the backpropagation algorithm [30] conducts a first-order gradient descent search in the weight space for the optimal weights to minimize the error rate [31]. Such a construction allows for what is referred to as deep learning,

where through a set of feedforward layers and backpropagation, feature weights are iteratively engineered, and interpretations of these values are difficult or even impossible for researchers to explain. As a result, neural networks like this that are used for classification and regression are something of a “black box” whose hyperparameters can be optimally configured but whose actual feature weighting need not be understood to be analyzed.

This thesis compares two different implementations of multilayer perceptron neural networks. The first MLP classifier is the last in this thesis to be implemented by the Scikit-learn library and was chosen to be representative of other simple neural networks as well as to be a comparison point for the second, more complex deep learning Keras/TensorFlow learning model. Keras is a high-level deep learning API that implements TensorFlow under the hood for the actual construction of the neurons and network [32]. TensorFlow is a multi-layered deep learning model much like the Scikit-learn MLP classifier but is constructed on a much lower level [33]. The Keras API allows for faster configuration of these TensorFlow neural networks, allowing quick configuration of deep learning frameworks and automatic validation testing. Keras/TensorFlow was chosen for this thesis due to its significant use in both literature and industry.

2.4 Challenges and Opportunities

Since 2005, many NBA video analytic research works have been proposed that have motivated and impacted the domain. This section contains a summary of the key challenges identified in the literature and highlights proposed solutions and directions for future research. The challenges and proposed direction are broken into three categories: (1) feature engineering and extraction, (2) deeper information discovery of actions, and (3) effective video content preprocessing and noise filtration.

2.4.1 Feature Engineering and Extraction

Machine learning solutions developed for traditional video- and image-based processing are not effective to provide NBA analytics using NBA video tapes. The main reason for this is that traditional video-based solutions aim to detect the variations of three main features: shape, color, and position. However, solutions for video-based NBA analytics aim to process videos by detecting multiple human actions executed simultaneously. Such data may have extremely similar instances in terms of shape, color, and position. However, effective features are generated by successfully detecting various player actions. A promising approach to address this challenge is to generate robust and more comprehensive features that analyze the context of human movement dynamics. Extracting and utilizing multiple detailed metrics and labels from videos is more appropriate than applying a single description [34].

2.4.2 Deeper Information Discovery of Actions

In NBA basketball, the same play can be executed in multiple variations. This means that the use of traditional metrics and distance functions (e.g., Euclidean distance) is not sufficient to simultaneously recognize similar plays and differentiate between different ones. Li et al. [35] showed that processing video frames within only a small temporal region is not effective for recognizing actions in videos. Moreover, they discovered that long-range dynamic information and deep features are essential for the discrimination of complex activities with shared sub-actions. This is a promising approach to adopt in NBA action and play recognition as it provides more detailed metrics that allow for more accurate similarity and dissimilarity estimations for actions and plays.

2.4.3 Effective Video Preprocessing and Noise Filtration

The nature of the data in this domain is unbalanced such that most of the game might not have any plays or actions of interest being executed. For example, Yu [6] discovered that only two minutes of on-ball screens (each taking about 1.3 seconds) occur in the entire forty-eight minutes of a standard game. This imbalance between events of interest and other events results in a ratio of approximately 4% relevant data. Without innovative and effective filtering of irrelevant data, the difficulty in identifying events of interest will continue to increase.

Algorithms that utilize rules designed manually by sport experts remain the sole method of filtering irrelevant content. Figure 12 shows an example of a rule that finds the ball handler (if any) [6]. If the algorithm returns NULL, then the frame can be discarded as no action (or play) is executed without the ball. Sampaio et al. [21] and Teramoto et al. [22] proposed using Principle Component Analysis (PCA) for data reduction. Also, Nistala and Guttag [3] propose using Convolutional Neural Network (CNN) to represent (approximate) the data in an abstract manner, which not only reduces the size of the data but also allows for more relaxed movement comparisons among actions and plays.

Figure 12. *Finding the Ball-handler Algorithm*

Algorithm 2: Finding the ball-handler

```
find ball-handler(video frame)
BH ← getmin(distance between Ball and players)
if distance between BH and Ball < 5 ft. then
 |  return H
else
 |  Return NULL
end
```

2.4.4 Annotation Scalability

Another challenge in this domain is the difficulty of obtaining a larger number of labeled events. This is because an annotator needs to watch a tape to judge when a certain play begins and ends and to classify what the play was. In addition to being a time- and labor-consuming approach, the beginning and end times of a certain action or play could be debatable, which impacts the effectiveness of machine learning approaches.

Despite the many research works and proposed solutions for action and play recognition in NBA data, many problems remain due to the complexity of recognizing and differentiating between actions and sub-actions of basketball plays. It is challenging to develop a unified framework with a machine learning pipeline that can filter, abstract, and label actions and plays in an accurate manner. Engineered features that consider contextual information about advantage creation, such as space created, defensive switches, and other player skillsets, are needed to train more accurate models. Additionally, utilizing and tailoring deep features to distill complex actions and strategies into sub-actions and sub-strategies is important for comparing actions with more accurate measures (i.e., fine-grain distance functions), such that those types of complex questions regarding on-ball screens can be answered. A unified framework would consist of a pipeline of supervised learning techniques to classify the on-ball screen dataset, a neural network and clustering approach to group the various approach and execution pairings present in that dataset, and a quality evaluation tool that takes into account features of advantage creation, such as space created or defensive switches favoring the offensive players' skillsets.

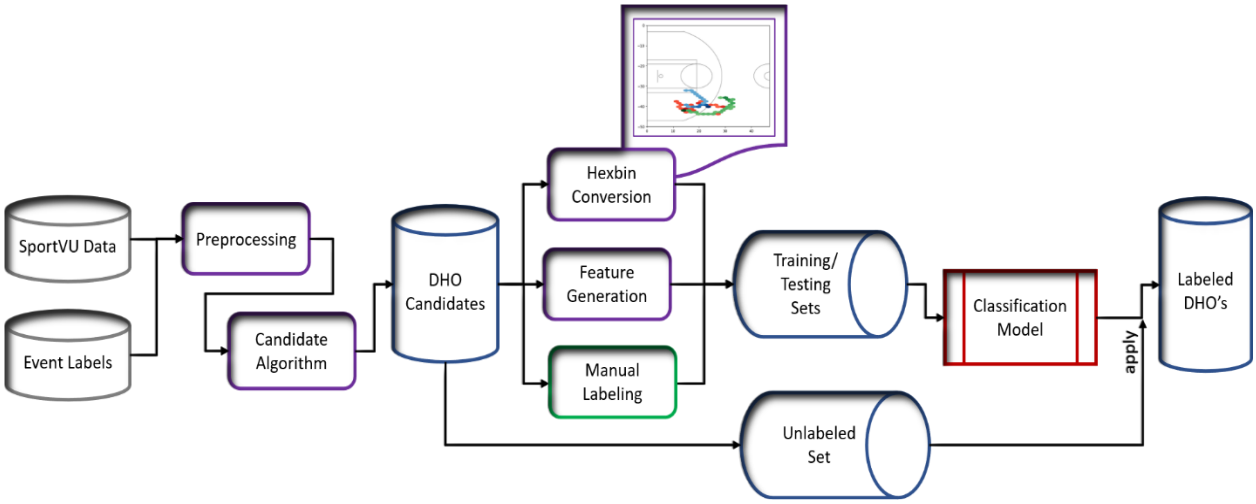
Chapter 3. Design and Implementation

3.1 Approach Design and Architecture

The pipeline presented in this thesis was designed to take in positional data and event labels representing the on-court actions over the course of a particular NBA game and produce a training set and model capable of classifying particular action instances. Although the focus of this thesis is on the DHO, and the end point for this pipeline is a trained classification model, the pipeline was designed in such a way that it can be adapted to support various play-actions and facilitate additional pattern mining or quality evaluation modules on the resulting dataset from the trained classification model.

To accomplish this, a sequence of preprocessing steps to combine and filter noise from the datasets were performed. The relevant games, players, teams, events, and moments are then serialized and stored in a relational database. To account for the variant nature of NBA actions, the pipeline implements a candidate algorithm to cast a wide net and collect as many action instances from the raw data as possible. These candidates are then reviewed manually by watching live game recordings. The positions of the screener, cutter and ball are then encoded at points of interest using hexbin maps and are included with the other extracted and engineered features. Once the features are complete, the resulting dataset can be split and used for model training and testing. A visualization of this process is found in Figure 13. Following is a more detailed description of each phase of the pipeline.

Figure 13. *The Dribble-hand-off Classification Pipeline*



3.1.1 Preprocessing

In the preprocessing stage, the two datasets (SportVU & Event Labels) are loaded in from their respective JSON and .csv files and combined by event id. Data for the teams and players involved in the game, such as the names and positions of players and the team cities and abbreviations, is then extracted and used to create the corresponding database entities for reuse.

Once the descriptive data is extracted and the datasets are combined, the events are narrowed to only those related to a successful or failed scoring attempt, a personal foul, or a turnover. This is because many of the events contain no corresponding positional data or because those events do not contain moments of interest in which a DHO or other action might occur. Examples include the events marking the start and end of a quarter and those corresponding to free throw attempts. Once this step is completed, the other non-pertinent data columns are removed, and event ids are generated so they can be stored in the relational database.

Finally, to complete the preprocessing steps, several higher-level features are extracted from a combination of positional data and event labeling to enable our candidate algorithm to

detect potential DHO actions. First, it must be determined which team is in possession of the ball for each given event. The raw coordinate data often extends well before and after the event in question, even overlapping neighboring events. This makes it difficult to tell which sections of the coordinate data ought to be ignored for a given event. Once possession has been determined, directionality can be established by looking to see which basket each team is attempting to score on (this then flips during the second half of play). When the possession and directionality have been confirmed, the moments for the event can be extracted from the raw data, assigned an id, and added to the database.

Throughout the preprocessing and subsequent processes, there were several cases where inconsistencies that could not be reconciled were discovered. An example of this is when a tracking camera switches identifications or loses a player entirely, resulting in players jumping wildly between two different locations. Other discrepancy cases where the clock suddenly reverts or becomes constant during play or even suddenly becomes a series of “NaN” values make it impossible to confirm based upon the raw data what exactly happened during the event. Considering the dataset is publicly available and from 2016, this thesis assumes that many of these inconsistency issues in the raw data collection have been addressed and are therefore of little academic interest to the machine learning applications present in this pipeline. Thus, odd events containing egregious raw data inconsistencies were removed from consideration.

3.1.2 Candidate Algorithm

Upon completion of the preprocessing stage, the focus shifts to identifying potential DHO candidates within the event data. To train the models, a collection of both positive and negative action examples must be gathered. This is not a trivial task, as it requires identifying, recognizing, and evaluating a series of different in-game states to determine the nature of the

possession and player actions. The candidate algorithm is designed to parse the preprocessed data and return a list of potential actions it has identified that might be instances of DHOs. This stage is discussed in more detail in section 3.2.

3.1.3 Manual (Human) Labeling

To manually label the candidates identified by the candidate algorithm, live broadcast recordings of the 15 games were obtained. Each game was then examined from start to finish, looking for example of DHOs, marking all the candidates as negative or positive and recording any actions missed by the candidate algorithm. The most commonly missed examples were atypical actions in which an unusually long pass occurred. In other cases, the screener extended the ball far past themselves, so the pipeline no longer registered them as being in possession of the ball, since the player tracking does not extend to limbs. An overview of the labeled data set can be found in section 3.2.3.

3.1.4 Hexbin Mapping

One challenge in pattern mining and extraction is overcoming the specificity of the data to identify more general trends. When using coordinate data and trajectories as precise as those contained in the SportVU dataset, similar actions may appear more distinctive than they are. In examining patterns in on-ball screen actions, many attempts have been made to abstract details from the coordinate data to better detect trends, often involving convolutional neural networks. These algorithms take in images of the players movements plotted onto a basketball court and produce fuzzy images that remove the precision of the data without compromising the representation.

This thesis seeks to produce a similar form of precision abstraction to efficiently train machine learning classifiers by using hexbin mappings of player trajectories instead of a CNN.

These plots are essentially hexagonal trajectory cell paths that embed the movements of a player or the ball in a series of hexagons color-weighted by time spent in that cell. Using these hexagon cell indices as opposed to extremely precise coordinates preserves the high-level patterns present in the data while abstracting away the details of the trajectory path. This process is discussed in more depth in a later section 3.3.

3.1.5 Feature Generating

Effectively training a classification model requires a descriptive feature vector that encapsulates the distinguishing aspects of a training example. Powerful classification models have been constructed on only a handful of engineered features. Therefore, this thesis assembles a selective list of fifty extracted and engineered features that seek to capture the characteristics and roles of the involved players' movements, as well as aspects of the wider state of play. Once these features are generated for a candidate, they are assigned an id, associated with the corresponding candidate, and added to the relational database. An exhaustive list and further discussion of these generated features is contained in a later section .

3.1.6 Model Training

The training models implemented in this thesis all use training and testing sets to fit the dataset and configure their predictions. While the specific configurations differ slightly, most models use an 80/20 training and testing split, and the Keras/TensorFlow model also uses a validation set to avoid overfitting. This is done to evaluate how each of the models performs when exposed to unseen data. Each classification model, especially the deep learning versions, can develop a prediction model from a dataset, but this is often achieved by learning specific trends and characteristics present in that dataset that might not scale to further examples.

Ensuring that model training always occurs on a disparate dataset ensures the models are configured to be as flexible and predictive as possible when exposed to new data.

The data were also normalized using min-max scaling to ensure that different scales did not create improper feature weights. The process of min-max normalization places all numeric features in a range from 0 to 1, with 0 being the min and 1 being the max. This ensures that a feature that typically falls between 0 and 1 is not overshadowed by a feature whose range is between 1-1000.

3.1.7 Model Testing

Once the learning models have been constructed on the training set, the remaining testing set can be used to gather accuracy and performance metrics to evaluate the quality of the model and feature set. The skewness of positive to negative examples means the data must be split carefully to ensure there are enough positive cases in the testing set for proper evaluation.

The model is tested by using its configured predictive model to generate a set of class designations for the testing set. Those designations are then compared to the actual labels of those examples, and four metrics are collected per class: false positives, false negative, true positives, and true negatives. These four metrics comprise what is referred to as confusion matrix and are the underlying statistics for many of the standard accuracy metrics, such as precision and recall.

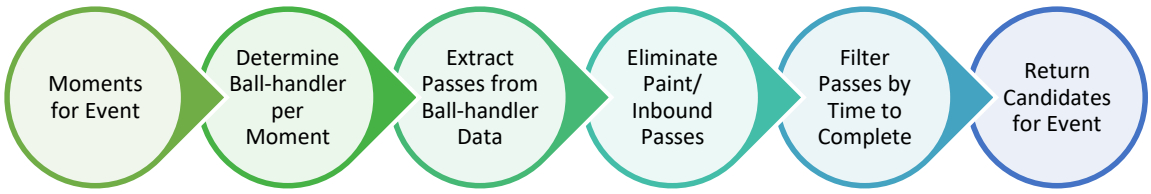
Models that achieve higher counts of true positives and true negatives can be said to perform better, although there are contrasting considerations. It is often difficult to make a model that effectively avoids false positives and negatives. Rather, it can typically only be skewed to be more conservative or aggressive in its classification, resulting in either fewer or more false

positives respectively. In cases where a false negative is more costly, for example, a cancer diagnosis, a model is skewed to err on the side of caution by designating a positive class if unsure.

3.2 Candidate Algorithm

As discussed previously, a rule-based algorithm can identify and extract basic actions from the complex, real-time player motion data. For example, a screen is often identified in the literature by calculating relative distances for all the offensive players on the court and looking for any pair whose relative distance is within a certain margin, often five feet. Obviously, such an approach is somewhat limited, because it is often the case that two players are within a five-foot proximity of each other, but no screen occurs. They may be jockeying for rebounding possession, running past one another, or simply standing closely. This makes classifying any action, even one as simple as a screen, a complicated task that requires either a great many specific rules that run the risk of overfitting the data or a small set of looser rules that either capture too much noise or fail to identify numerous positive instances.

Figure 14. *Steps in the candidate algorithm selection process*



Due to these challenges, relying only on rule-based algorithms to identify and classify classifying complex actions involving multiple coordinating players provides limited outcomes, especially in terms of precision. The candidate algorithm is therefore designed with a minimal

rule set intended to capture a wide training set of both positive and negative instances so that the actual classification can be performed by a machine learning model informed by the actions that pass the candidate algorithms rules.

3.2.1 Identifying Passes

Much like in the example from the literature attempting to classify on-ball screens, the first step in the dribble-hand-off classification pipeline is to identify who is in possession of the ball at a given time (see Figure 13). In basketball, this individual is referred to as the ball-handler and can be found by taking the Euclidean distance from the ball to each of the offensive players and choosing the player corresponding to the minimum distance. While this is not perfect and cannot capture certain situations, like a player fumbling a ball, it is a reasonable start given the constraints of the dataset. A few additional rules concerning the radius of the ball are added to help rule out instances where a shot is taken, or players are fighting for a rebound and no one is technically in possession. Finally, a distance threshold of five feet is set, and for all moments in which no offensive player is under that threshold, possession is marked as "N/A".

From the ball-handler data, a list of all passes for a given event can be extracted by looking for shifts in possession. Since the moments when possession is lost and regained are marked and associated with a specific coordinate in the SportVU data, this data can be used to set the start and end locations of the pass and determine how long it took for the pass to be completed.

3.2.2 Selecting Candidates

Finally, the list of extracted passes is combined with two additional rules to eliminate certain types of passes, such as those that occur in the painted area closest to the basket (the paint) or those that originate from out of bounds. While it is not impossible to have a DHO that

occurs in the paint, these additional rules filter a significant amount of noise from the final dataset, while losing only a handful of positive instances.

The passes are then evaluated by how long it took for the pass to be completed. This is measured by looking at when the ball first leaves the possession of the passing player and the corresponding moment in which the ball first enters the possession of the receiving player. Any remaining pass that occurs within a small enough time span (typically eight moments, depending on the capture rate for that particular game) is nominated as a candidate. The candidate is then assigned an identifier and a label, and a list of identifying features is created for the manual labeling process before it is added to the relational database.

3.2.3 Evaluating Selected Candidates

Of the resulting 1811 candidates selected across the fifteen games reviewed in this thesis, a total of 568 were identified as positive DHO instances. A portion of the positive DHOs, ~4.6%, were marked as fakes, which are discussed in depth in section 1.3.1. A summary of the breakdown for the labeled candidates, as well as the associated source data, can be found in Table 3:

Table 3. *A Summary of the Total Entities Present Within the Training Set*

Thesis Data Overview			
Games	15	**Total Candidates**	1811
Teams	18	**Positive Candidates**	568
Players	241	**Negative Candidates**	1243
Events	1264	**Fake Candidates**	26

Ultimately, the intent of the candidate algorithm is to cast a wide net and collect as many action instances as possible that can later be narrowed by the trained classification model. Therefore, time was not spent further engineering or optimizing the rules that dictate the

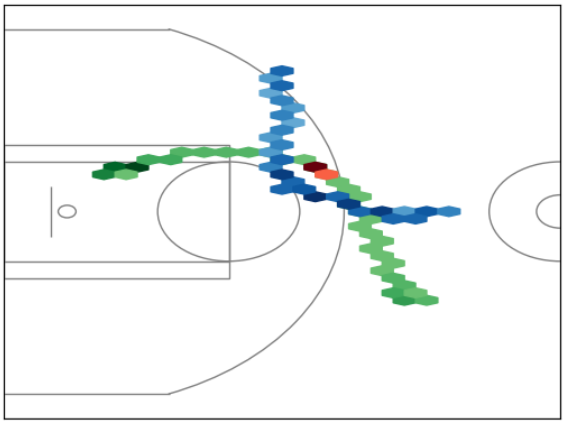
nomination process. A total recall of ~90% was achieved during manual labeling, and this was deemed sufficient to train the model. Additional steps to increase recall or narrow the scope of candidates can be considered in future work.

3.3 Hexbin Positional Conversion

One challenge in identifying patterns in such a diverse dataset is overcoming the granularity and precision of the raw data. Each positional coordinate in the SportVU moments contains two floating point numbers with as many as eight trailing digits. Feeding this raw data into a classifier can be confusing, as two nearly identical positions may evaluate to have low similarity scores.

To abstract away some of these play/action details and capture the higher-level descriptive elements of players' court positions at key moments during an action, this thesis uses a technique known as hexbin mapping to embed player trajectories into cell paths. Binning occurs by separating a continuous data into discrete sections, in this case hexagons. The raw coordinates are then placed into each one of these bins, and the resulting occupancy counts of those bins can be used to represent the original data. An example of hexbin mapping in Figure 15 shows how this strategy can be used to convert the trajectory paths for the screener (blue), cutter (green) and ball (red).

Figure 15. *Hexbin Mapping of Coordinate Data for DHO Action*



In this thesis, these hexagon positions are used as features in place of the raw coordinate data. This abstraction means that two actions that begin in nearly the same exact location will appear that way to the classification models. Other efforts to abstract away these specific details in order to identify more general patterns have been used to explore NBA strategies in the literature before. This is often done by creating trajectory images of the raw data and feeding them into a convolutional neural network and clustering those results. To the best knowledge of this thesis, however, this is the first attempt in research to embed player positions using hexbin mapping for the purpose of DHO classification.

3.4 Feature Creation

A selective list of fifty features was generated for each candidate. A total of 16 of these features are engineered, in that they require significant manipulation or composition past the point of a simple calculation. The remaining thirty-four features were extracted from the raw dataset and may or may not have gone through one or more processing steps amounting to basic calculations. Once all the features were collected, they were combined with the manual label,

given a derived id, and stored in the relational database for use in training the classification models. Both the extracted and engineered features are discussed in detail below.

The features generated were derived from two-second windows before and after the screen moment, defined as the moment when the coordinating offensive players are closest together. By distinguishing the approach and execution stages within the feature set, examples with vastly different executions can still be classified similarly by the classifiers.

3.4.1 Extracted Features

In total, forty-two of the features used to train the classification models for this thesis were extracted from the combined SportVU and event labels dataset. Most of these extracted features required some basic processing to derive meaningful metrics from the data before being passed off to encoding and normalization steps. The exceptions to this are the player archetype and the ball radius, which is a floating point measure of the ball size on the camera that indicates the approximate height of the ball, where larger radius measures indicate a greater height along the z-axis. The ball radius is left as a floating-point value and passed along to the normalization process, but the player archetypes require some form of encoding because it is difficult for a classification model to compare string values. Encoding the player archetype designation creates an enumeration which in turn can be represented by an integer and accurately utilized by the learning model.

The remaining thirty-six extracted features consist of various measures of the paths traveled by the cutting and screening players as well as the ball itself. First of which are the distances each traveled, as well as the relative distance of the players at key moments of considerations: the beginning of the approach stage, the pass moment, the screen moment, and the end of the execution stage. These are the same moments for which the engineered location

features discussed in the following subsections will be calculated. In addition to the distance measures, the average speed of the cutter, screener, and ball as well as the slope and trajectory of their plotted movements over the action duration are derived from the raw data. A full overview of these features and their descriptions can be found in Table 4 below.

Table 4. *Overview of Extracted Features (42 total)*

Feature	Feature Type	Feature Description
screener archetype	Enumeration	Encoded string containing players' assigned positions
cutter archetype	Enumeration	Encoded string containing players' assigned positions
distances traveled*	Float	Calculated for the screener/cutter/ball
average speeds*	Float	Average speed of screener/cutter/ball
trajectory slopes*	Float	Slope of screener/cutter/ball trajectory
trajectory intercepts*	Float	Intercept of screener/cutter/ball trajectory
ball radiuses**	Float	Radius of the ball indicating relative height
relative distances**	Float	Relative distance between the screener and the cutter from each other and from the ball
***** *represents multiple features collected for the cutter/screener/ball from the approach and execution stages*		
****** *represents multiple features collected from the start/end of the action and the screen/pass moments*		

3.4.2 Engineered Features

In this thesis, a total of sixteen features required significant enough abstraction or data manipulation that they were considered engineered. The first of this set, the hexbin cell transform of the location data, has already been discussed in detail. While the other engineered features in this thesis would likely be found in sophisticated classification efforts for different actions, they have not, to the knowledge of this thesis, been used to distinguish DHO candidates.

These features consist of higher-level composite features that, when combined with the raw coordinate data and a rule-based algorithm, contribute a deeper contextual and temporal understanding of the on-court behaviors present in the action. The first of these measures is the offset into the developing play itself. This is based off of the insight that coordinated offensive

actions typically require time to set up, execute, and then create a scoring opportunity. This means such actions are less likely to occur at the very start of the possession, when the majority of offensive players are likely still on the far side of the court, nor at the very end of a possession, when players are scrambling to get up an attempt before committing a shot-clock violation. Similarly, all ten player locations are considered to determine how many are on the half of the court where the team in possession is currently trying to score. If only seven of the ten players are past half-court, the players are likely in a fast-break offensive situation in which each player is darting towards the rim or three-point line, and the potential for any type of on-ball action is significantly limited.

Finally, in addition to the duration of the pass (derived from the pass detection algorithm described in the candidate algorithm), the initial pass is examined to detect if it originated from out-of-bounds, which indicates what is referred to as an inbounds play. Inbounds play are unique offensive opportunities in basketball, in which players and often coaching staff are permitted to gather and discuss strategy ahead of the proceeding play. This opportunity for verbal collaboration commonly results in a higher potential for coordinate team action and, therefore, an increased rate of play-actions such as DHOs. Ultimately, these engineered features are calculated for each candidate alongside the extracted features and combined along with an id and manual label to create the final testing/training set. An overview of each of these engineered features can be found in Table 5.

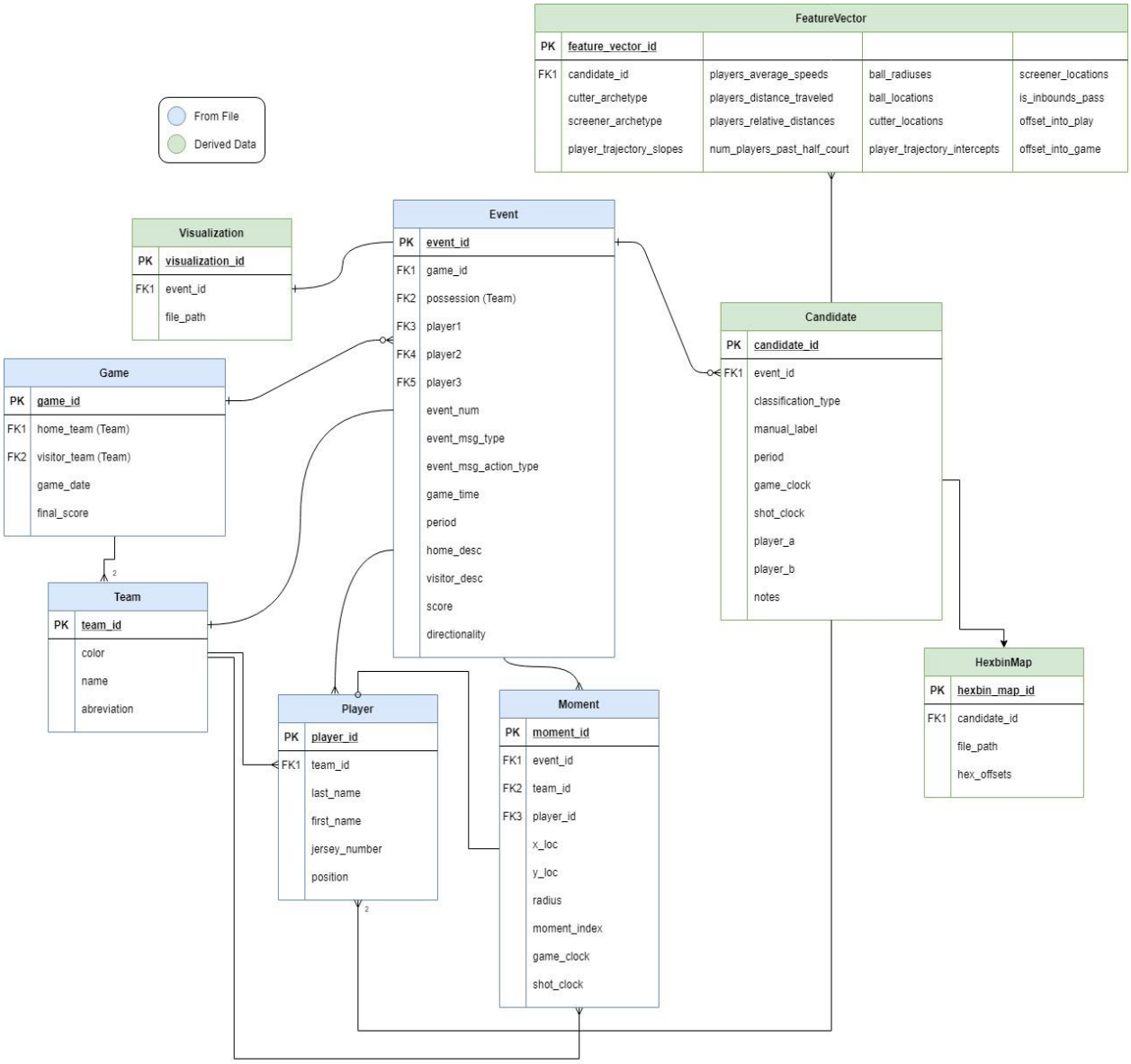
Table 5. *Overview of Engineered Features (16 total)*

Feature	Feature Type	Feature Description
cutter locations*	Hexbin coord.	Coordinate of closest hexagon cell for cutter on the start/end of the action and pass/screen moments
screener locations*	Hexbin coord.	Coordinate of closest hexagon cell for screener on the start/end of the action and pass/screen moments
ball locations*	Hexbin coord.	Coordinate of closest hexagon cell for ball on the start/end of the action and pass/screen moments
offset into play	Integer	Indicates which sixth of the play the action occurred in
pass duration	Integer	The total amount of moments captured between the start and end of the pass
players past half court	Integer	Count of players currently on the same side of the court as the occurring action
inbounds play	Boolean	Indicates whether play is initiated from an out-of-bounds pass
***** *represents multiple features collected from the start/end of the action and the screen/pass moments*		

3.5 Entity Overview

As discussed in previous sections, accompanying the classification pipeline is a relational database constructed to represent the various entities of interest and store all the data required to generate the features and train the machine learning models. Serializing the data as it progresses through the pipeline also allows for future access to occur through the database models, improving overall performance. Considering the vast size of the dataset, even for a 15-game sampling, computational complexity with large datasets is an inherent challenge in any developed pipeline. The overall database for this thesis alone consists of more than five million rows across the relational entities overviewed in Figure 16:

Figure 16. *Entity Diagram of the Classification Pipeline Domain*



Chapter 4. Evaluation

4.1 Accuracy Measures

To evaluate our approach, we use several performance metrics to investigate its accuracy and completeness under different settings. The most popular accuracy measures in machine learning and information retrieval domains are accuracy, precision, recall, and F_1 score. Following is a presentation and explanation of each measure where true positive (*TP*) is the number of dribble hand-off candidates that are correctly classified as true. True negative (*TN*) is the number of instances that are correctly classified as false. False positive (*FP*) is the number of instances that are incorrectly classified as DHOs. False negative (*FN*) is the number of instances that are incorrectly classified as not being DHOs when, in fact, those instances were positive examples.

Accuracy (A): calculates the number of correctly classified instances (negative and positive) over the total number of instances, using the following equation:

$$\mathrm{A} = \frac{(\mathrm{TP} + \mathrm{TN})}{(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN})}$$

Precision (P): calculates the number of correctly detected class members by the classifier over the total number of correctly and incorrectly detected members using the following equation:

$$P = \frac{TP}{TP + FP}$$

Recall (R): calculates the number of correctly detected class members over the total number of class members using the following equation:

$$R = \frac{TP}{TP + FN}$$

***F_1* score:** provides a score that balances both the concerns of precision and recall in one measure using the following equation:

$$F_1 = \frac{2 \times (precision \times recall)}{(precision + recall)}$$

In addition to these metrics, receiver operating characteristics (ROC) were generated for each of the learning models. An ROC is used to compare the false positive and true positive rates of a model. A completely random model with an even data distribution tasked with choosing between two classes can be expected to do so with approximately 50% accuracy. If plotted, this would result in a diagonal line at a 45-degree angle extending out from the origin. The area under that curve (AUC) would be 0.5, or approximately 50% of the domain space. A model that efficiently distinguishes between class designations, on the other hand, would perform much better and, in turn, have a much higher AUC. It has been found that an AUC score of closer to 1 not only indicates a highly performant and predictive model, but that such a measure is even more telling than accuracy metrics [36].

To provide that additional context, this thesis provides an ROC graph for each of the classification models, as well as learning curves that indicate the overall fitness of the model as it exposed to an increasingly large training set. By looking at both of these graphs in combination with the accuracy metrics on unseen data, an overall effectiveness for these classification models and their similarly constructed relatives can be ascertained.

4.1.1 Metric Collection Methodology

In this thesis, all reported metrics were derived from many individually configured models in order to obtain an unbiased and representative measure of performance for each type of learning model presented. For each model, the training and testing splits were separated from

the full candidates, the model was fit on the training data, and a set of predictions were made for the testing data. Those predictions were then evaluated for the performance measures accuracy, precision, recall, and F_1. The variance in class representation within the training set will result in different interpretations of the data resulting in different predictions. Averaging metrics across many different splits of the data removes the bias that may result from a single model and a single sample, which could significantly under- or overperform typical performance.

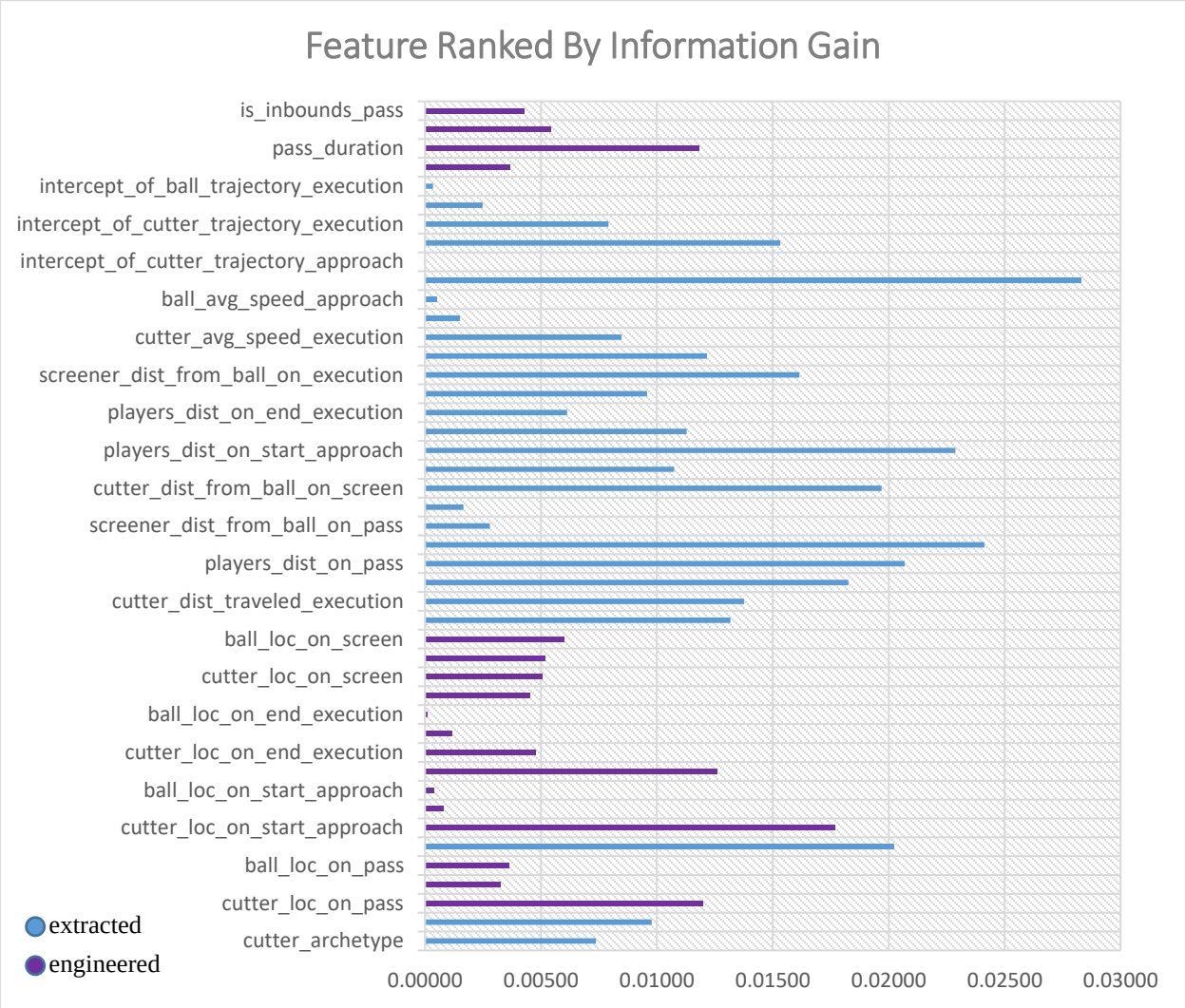
4.2 Evaluating Features

Before examining the effectiveness of the machine learning algorithms, the features themselves are examined. There are many ways to evaluate features and measure their importance to the classification model and its accuracy. Some of these methods (principal component analysis, univariate feature rankings and more) have already been used in the literature. In this thesis, features are examined using information gain as a measure to assess not only which features offer the most predictive value but also how the engineered features compare to the extracted features.

4.2.1 Information Gain

There are several metrics available to evaluate the quality of a feature set. Considering that the effort of this thesis is to build a classification model, information gain is an attractive method, as it ranks features in relation to some targeted dependent variable. Information gain is determined by calculating the mutual information of each variable in the feature set with respect to the classification based upon the entropy of those elements [37]. The resulting rankings indicate which features have the most predictive power within the dataset. A summary of the feature rankings for this thesis is found in Figure 17.

Figure 17. *Overview of Information Gain per Feature*



While most features have reasonable scores, and the location of the ball and cutter when the screen occurs stand out in particular, the most interesting takeaway here is the distribution of extracted and engineered features. Most of the features in this thesis are extracted and involve derived calculations of the movements of the players and the ball. Considering the diversity of the base dataset and the wide representation of actions and variants, it is reasonable that such a set of features would be necessary for accurate classification. The performance of the engineered features compared to the extracted features, however, is a strong indication of their descriptive

capability. Ultimately, the intention of an action classification feature set is to provide a concise and informative description of the data that ideally captures aspects of the larger context and intentions of the involved players. The propensity for information gain present in the engineered features of this thesis demonstrates a capability to do just that.

It should be noted that while some features have very low or zero scores for information gain in this sampling, it was discovered that removing these features from the classification experiments resulted in lower performance in terms of recall, precision, and accuracy. This indicates that despite the results from the information gain algorithm, preserving those features provided a marginal but nonnegligible benefit.

4.2.2 Impact of Hexbin Mapping on Model Performance

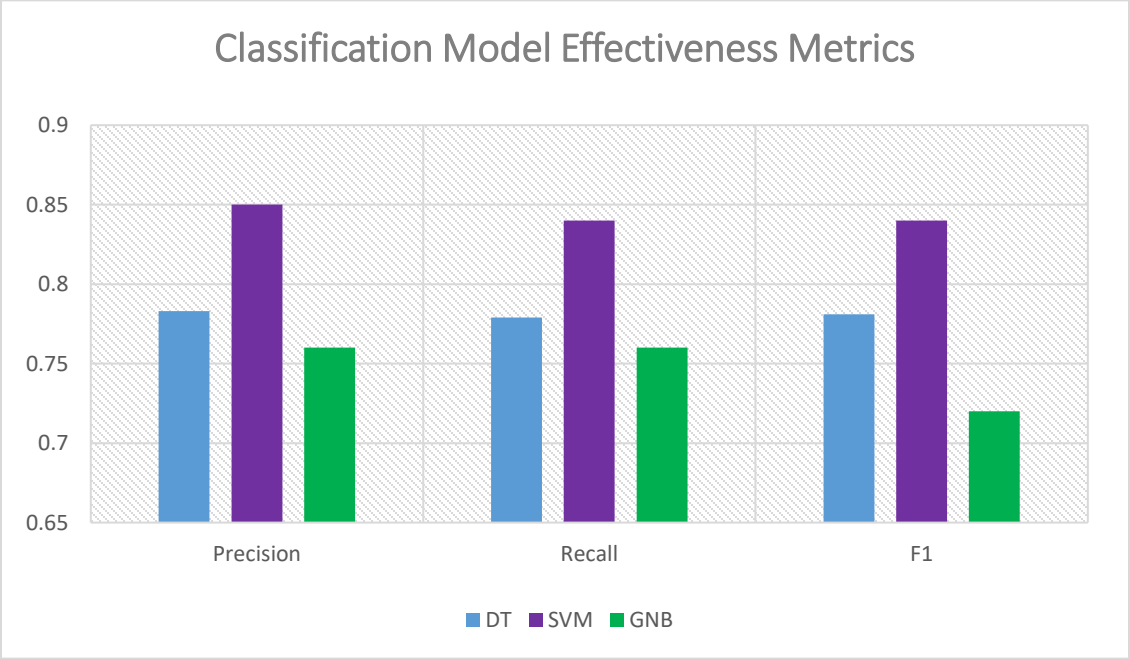
To determine the effectiveness of the hexbin mappings, a training set was generated with the raw coordinate encodings. Both sets of coordinates were sorted and encoded for ingestion by the classification models. The hexbin coordinate set was ~40% of the size of the raw coordinate set, indicating a significant reduction in precision. Because the training set a model receives has a large impact on its accuracy, both training sets were used to train an MLP network, and the results were averaged. While the raw coordinate set still performed well, it was ~1% less accurate with ~3-4% lower AUC. This is a significant but not substantial margin that has potential to increase with a larger more robust training set. It is the opinion of this thesis that while hexbin mapping has potential to improve a learning model's ability to classify, the bigger impact is still likely to be found with image based deep learning as discussed in section 2.2.2.

4.3 Comparing Classification Models

To evaluate the effectiveness of the constructed pipeline and composed feature set, the training data was first fit on and subsequently evaluated by three classification learning models

implemented by the Scikit-learn library: support vector machine, gaussian naïve bayes, and decision tree. SVM is a favorite among the literature for the classification of other on-ball actions and shows promising potential in classifying DHO candidates as well. All models in this thesis were configured independently and then assessed by averaging results from 10 distinct iterations with unique testing split to avoid placing too much stake in any individual model. As seen in Figure 18, SVM outperforms the other two learning models on precision, recall and F_1 score.

Figure 18. *Comparing Accuracy Metrics Across Classic Classification Models*



These three metrics are the basis for much of our comparison, since they evaluate not only how accurate our model is but also how many of the positive instances in the dataset were identified and how much extra noise was misclassified as a part of this effort. These statistics are derived from the total number of correct classifications as well as false positives and negatives. A total breakdown of how each model performed is seen in Table 6.

Table 6. *Confusion Matrices for Classic Classification Models*

		Negative	Positive
Decision Tree (DT)	Negative	321	65
	Positive	55	102
Support Vector Machine (SVM)	Negative	341	51
	Positive	35	117
Gaussian Naïve Bayes (GNB)	Negative	371	15
	Positive	114	44

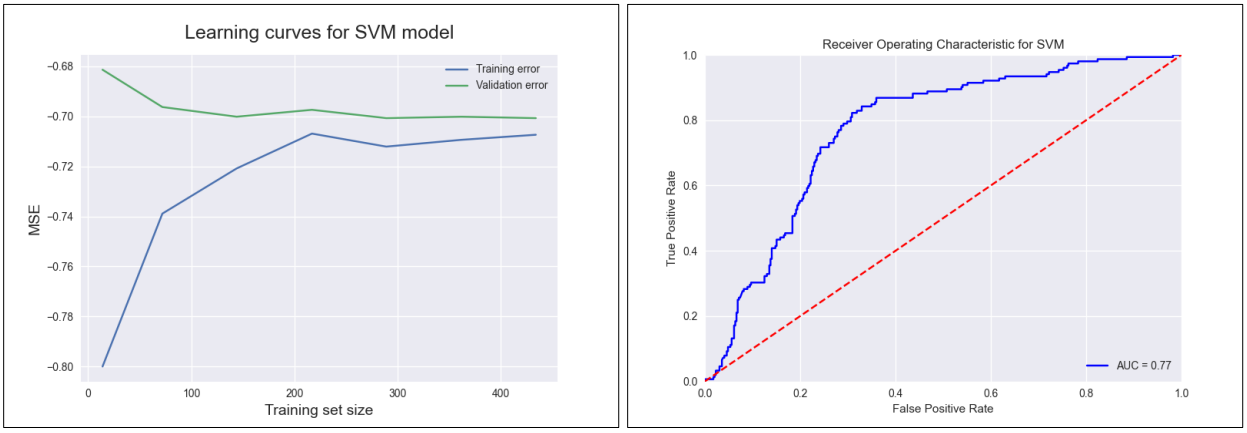
While these indicators seem to suggest that SVM is a clear favorite for classification among these options, another measure of a model's fitness, the ROC curve, offers additional insight into how well the model discerns between the classifications. This is important, because while our training set is substantial, it is still a relatively small sampling compared to the larger context in which thousands of games are played in the NBA in a single year. Training our models on a smaller sample, therefore, can result in limitations on performance. The ROC curve suggests the upper limit for a model's performance given access to a larger training set [36].

4.3.1 Support Vector Machine

SVM performed quite well on the dataset, reaching an average accuracy of around 85%. Considering the diverse and stochastic nature of the actions and the case that a fair number of the variants are fakes intentionally executed to be misleading, this is a promising result. As seen in Figure 19, the training and validation error curves converge fairly quickly, indicating a potential

for overfitting that possibly requires additional training examples or a smaller learning rate. Perhaps most interesting is that while SVM outperformed the other two classification models in accuracy metrics, its AUC was less impressive. This might suggest that while early returns are promising for SVM, the other learning models might be more robust and better respond to a larger training set.

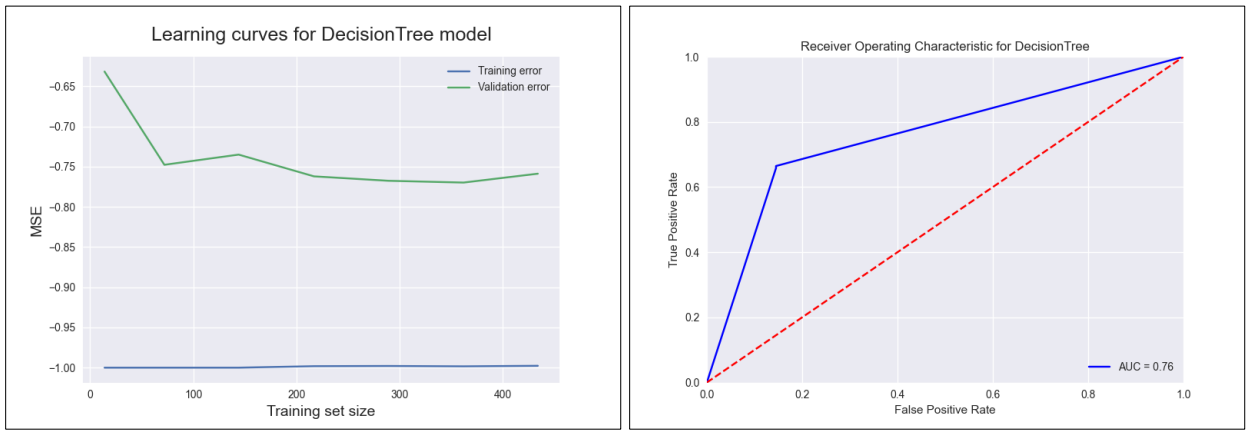
Figure 19. *Learning and ROC Curves for SVM Model*



4.3.2 Decision Tree

The decision tree classifier is not as suited to this diverse a data set. It underperforms compared to the other approaches and, despite attaining a comparable AUC to that of SVM, does not demonstrate much promise going forward for an action classification pipeline. The learning and ROC curves are interesting for the decision tree, as the model fits closely to the training data and, in fact, constructs its tree from it. As a result, training error tends to be quite low but converges with validation error at a lower accuracy than is desired. It is possible that in future work a more complex random forest could be implemented and fit on this data with better results, but based on the experimentation and results presented here, the DT classifier does not perform well at this task.

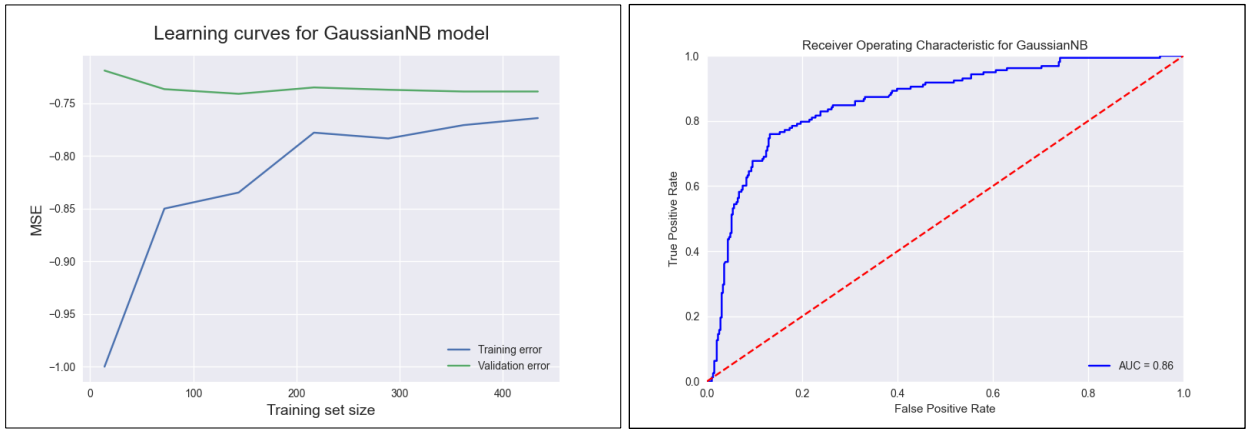
Figure 20. *Learning and ROC Curves for DT Model*



4.3.3 Gaussian Naïve Bayes

The last of the simple supervised learning algorithms examined in this thesis, GNB, shows the greatest potential for discernability. Although it did not score as highly on the accuracy metrics as SVM did, AUC was significantly higher at 86%. The tedious task of manually reviewing and labeling means that obtaining the nearly two thousand candidates studied in this thesis alone was difficult. A fully trained pipeline would likely need to be fed hundreds if not thousands of times as many examples. Therefore, the AUC is a promising sign for GNB, as it indicates that it might be a strong choice for a larger data set.

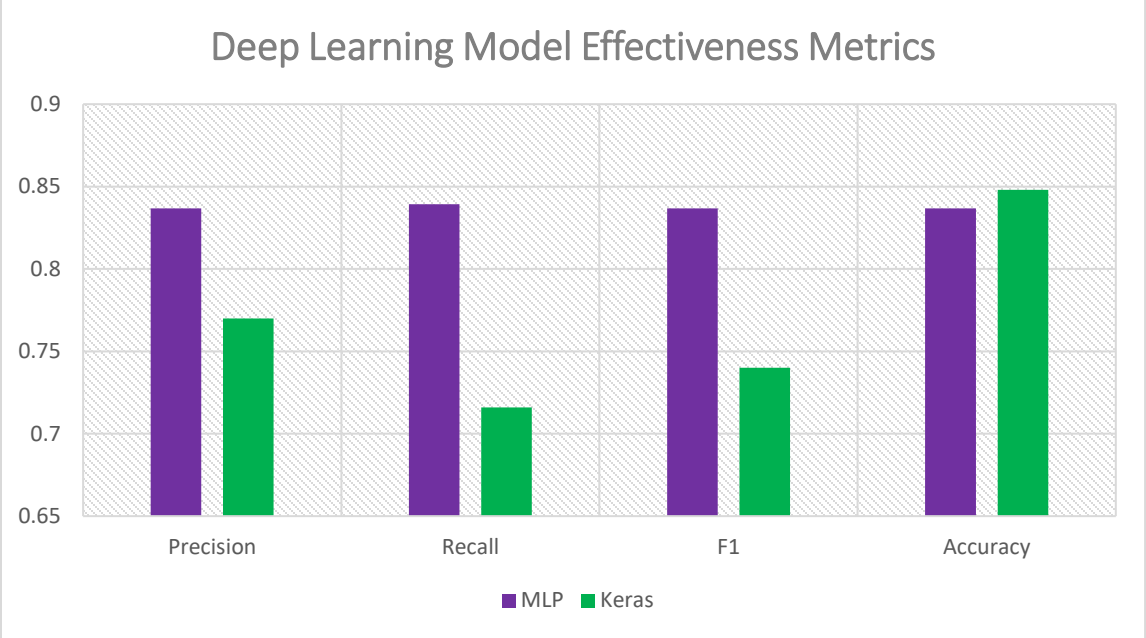
Figure 21. *Learning and ROC Curves for GNB Model*



4.4 Comparing Deep Learning Models

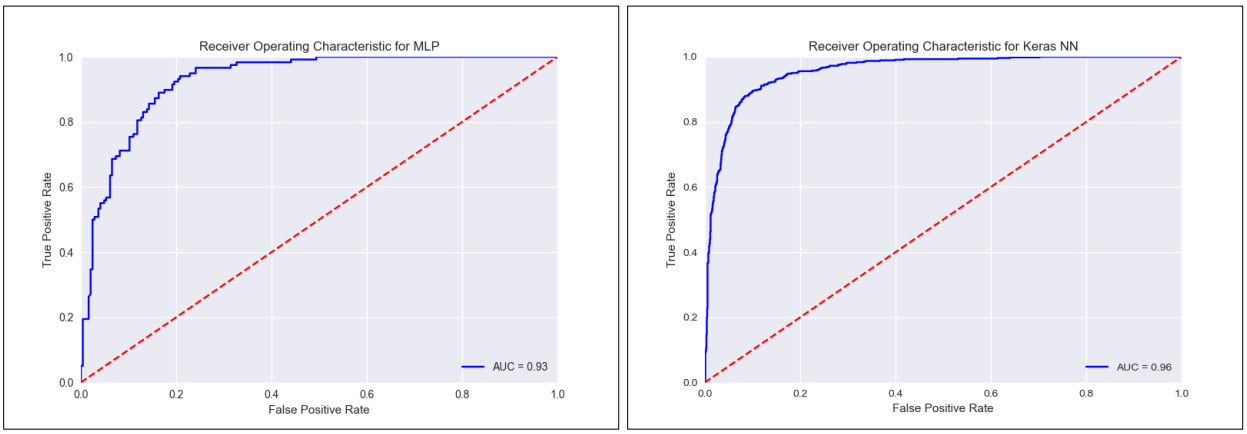
In addition to the three supervised learning approaches discusses above, this thesis implemented two neural networks to evaluate the effectiveness of deep learning on the DHO candidate dataset. As seen in Figure 22, neither the Scikit-learn MLP nor the Keras/TensorFlow algorithms significantly outperformed SVM in terms of accuracy, precision, nor recall.

Figure 22. *Comparing Accuracy Metrics Across Deep Learning Models*



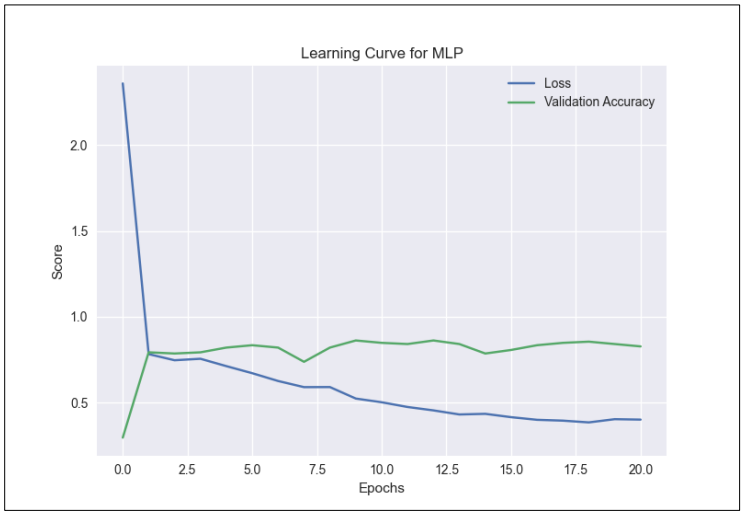
Both neural networks scored significantly higher on AUC, each topping 90%. This makes sense, as the appeal to deep learning is that the models themselves decide on how to weight and value features, often identifying and capitalizing on patterns that are difficult for human researchers to deduce after the fact. The ROC curves shown in Figure 23 indicate flexible and highly predictive models that have the potential for increased performance with increased data.

Figure 23. *Comparing ROC Curves of Neural Networks*



The actual training and configuration of these models is also more intensive than that of their simpler classification counterparts. The neural networks feed data forward through multiple hidden layers while backpropagating results to ensure optimal feature weighting. As a result, there is real risk of overfitting the model to the training set, so additional validation with unseen data during the training process is required to maintain the model's adaptability. Both neural networks implemented Adam optimization solvers and Relu activation functions for the hidden layer perceptrons [38]. Figure 24 shows the learning curve for the MLP classifier, which illustrates how loss and validation accuracy are configured across epochs.

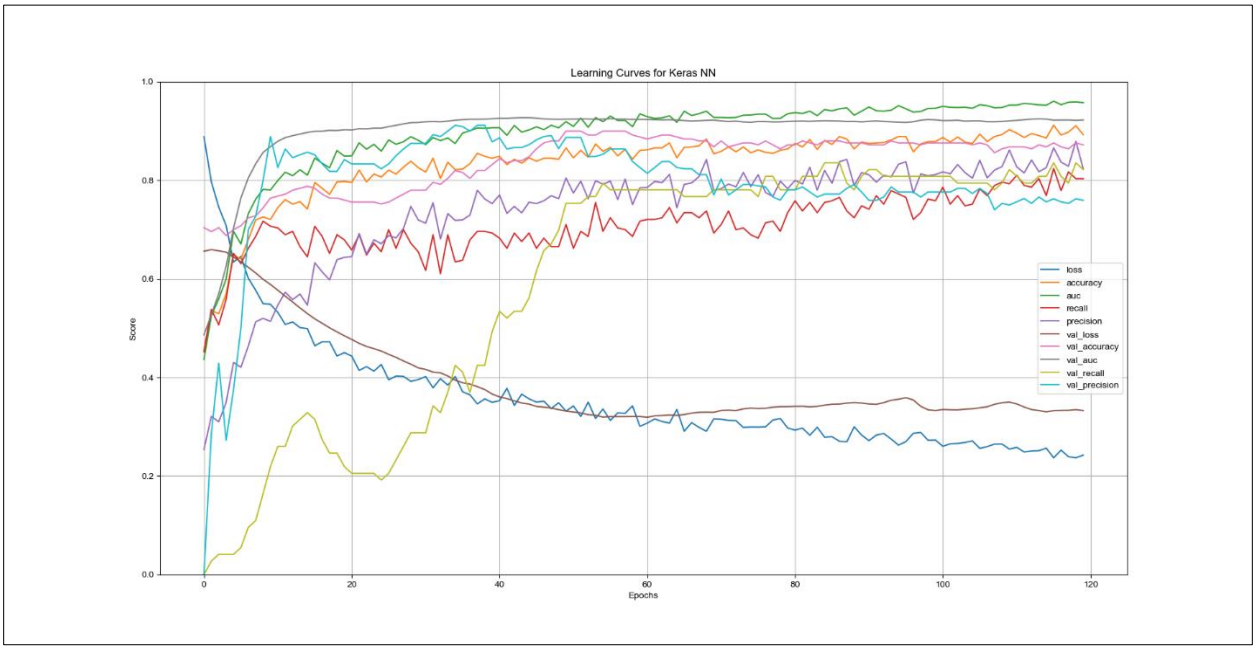
Figure 24. *Learning Curve for Scikit-learn MLPClassifier*



In an effort to avoid overfitting, the learning model will automatically stop when it detects that it has begun to converge, which occurs in this example at epoch 20. As the training set increases in size, this will likely take longer to occur, allowing for more efficient classification.

The Keras/TensorFlow MLP was also trained over epochs but required a longer time and more precise configuration. Keras neural networks optimize based on a variety of different metrics; those used in this thesis were accuracy, precision, recall, and AUC. Figure 25 shows a chart with the learning curve as each of these metrics converges over 120 epochs with a validation test size of 20%.

Figure 25. *Learning Curve for Keras Neural Network*



The Keras deep learning approach struggles to balance precision and recall when classifying unseen data, but it produced the greatest AUC score of any considered model, indicating strong potential for improvement given a larger dataset. With the training set currently present in the thesis, preventing the Keras model from overfitting was quite difficult, which

resulted in many more false identifications than the MLP classifier. A summary of the average confusion matrices for both classifiers is found in Table 7.

Table 7. *Confusion Matrices for Deep Learning Models*

		Negative	Positive
Multi-Layer Perceptron (MLP)	**Negative**	**220**	**33**
	Positive	**26**	**84**
Keras Neural Network (Keras)	**Negative**	**338**	**35**
	Positive	**46**	**116**

Chapter 5. Conclusion

This thesis proposed an end-to-end approach to automatically detect dribble hand-off actions. First, it surveyed approaches to NBA content analysis, the types of machine learning algorithms used to classify and extract patterns from player movement data, the challenges with feature engineering and noise filtration present in these pipelines, and the current solutions implemented in this domain. This thesis presents a constructed pipeline capable of ingesting raw data and producing a classified dataset of dribble hand-off actions in a number of approach and execution strategies, including fakes. Finally, it assembles a large manually-labeled dataset for training, explores new approaches in data representation and feature engineering within NBA action classification, and presents three supervised and two deep learning models for evaluation.

Ultimately, play-action detection, classification, and analysis are still in the early stages. While great strides have been made in assembling machine learning strategies to perform these tasks, the resulting work is mostly too limited to truly represent the intricate system that is a basketball game. By limiting the domain, it is easier to see how such an overall system might be assembled. Identifying and classifying basic actions is nearly a solved problem, and advances in deep learning show great promise for pattern mining such a complex system. It is the opinion of this thesis that there is a great deal of room to build out this pipeline specifically for other on-ball actions and to eventually branch out to other less prominent strategies. Once similar pipelines are constructed to analyze defensive and offensive strategies present among all ten players on the court, it may even be possible to completely automate the tedious work currently performed by staff videographers and coaches.

5.1 Discussion of Learning Models

Ultimately, in evaluating the machine learning models explored in this thesis, there are two clear standouts. Among the simpler classification models, SVM significantly outperformed DT and GNB. It even narrowly edged out the MLP classifier on precision, recall and F_1 score. The ROC curves for both of the deep learning approaches and GNB indicate that these models might outperform SVM given a larger, more diverse dataset. Future work to optimize the effectiveness of this pipeline could focus on building larger datasets, more descriptive features, and more narrowly targeted training examples. Considering the difficulty and novelty of the problem, this thesis concludes that effective and accurate classification attempts in this domain are an interesting but attainable challenge.

5.2 Cross-domain Application of this Approach

While the application of the techniques and learning models used in this thesis is specific to basketball strategies implemented by NBA teams, there are opportunities for similar patterns to be used in other domains interested in classifying and examining patterns in trajectory data. For example, an airline company that has flight data from thousands of different flights in and out of a single international airport. Models could be configured to evaluate those flights for total flight time, fuel costs, or any other dependent variable of interest. To facilitate such a pipeline, the trajectories of those flights could be converted to hexagonal cell paths using hexbins, and the encoded locations and flight paths could be used abstractly to interpret higher level patterns and features present in the data. In fact, any domain that is interested in studying patterns of trajectories or in reducing the precision and dimensionality of coordinate data for comparison might consider a technique similar to that presented in this thesis.

5.3 Future Work

It is the belief of this thesis that the action classification pipeline presented here is only a starting point for additional research and analysis. The generated hexagonal cell path images can be passed off to an unsupervised clustering module and mined for patterns in approach and execution variance. The results of these actions can be taken into consideration along with the spatial alignment of players and the concept of court realty to provide a contextual and non-outcome-based quality metric to compare actions [39]. The movements of defensive players and their strategies in defending dribble hand-offs could be plotted and explored in a similar manner. Other on- or off-ball actions could be added via new rules sets for the candidate algorithm and feature vectors for the classifiers to study different player movements. A natural language layer could be superimposed upon the final dataset to allow for high level querying and meta-analysis. As both basketball and machine learning strategies continue to develop, there will continue to be novel ways to interpret data, represent meaning, and train learning models. While the explicit contribution of this thesis is an end-to-end classification pipeline, the hope is that it provides a foundation for further action detection classification, pattern mining, and analytics research and that it inspires others in the domain to think creatively about how to embed the semantics of basketball into machine learning applications.

References

[1] A. McIntyre, J. Brooks, J. Guttag, and J. Wiens, "Recognizing and Analyzing Ball Screen Defense in the NBA," in *Proceedings of the MIT Sloan Sports Analytics Conference, Boston, MA, USA*, 2016, pp. 11–12.

[2] A. McQueen, J. Wiens, and J. Guttag, "Automatically Recognizing On-ball Screens," 2014.

[3] A. Nistala and J. Guttag, "Using Deep Learning to Understand Patterns of Player Movement in the NBA," in *Proceedings of the MIT Sloan Sports Analytics Conference*, 2019, pp. 1–14.

[4] N. Seward, "Building an On-ball Screen Dataset Using Supervised and Unsupervised Learning," University of Ontario Institute of Technology, 2018.

[5] K.-C. Wang and R. Zemel, "Classifying NBA Offensive Plays Using Neural Networks," in *Proceedings of MIT Sloan Sports Analytics Conference*, 2016, vol. 4.

[6] A. Yu and S. S. Chung, "Automatic Identification and Analysis of Basketball Plays: NBA On-Ball Screens," in *2019 IEEE International Conference on Big Data, Cloud Computing, Data Science \& Engineering (BCD)*, 2019, pp. 29–34.

[7] J. D. Brooks, "Using Machine Learning to Derive Insights from Sports Location Data," Massachusetts Institute of Technology, 2018.

[8] S. Narayan, "Applications of Machine Learning: Basketball Strategy," Massachusetts Institute of Technology, 2019.

[9] "STATS SportVU®Basketball Player Tracking." .

[10] K. Saini, Udam; Evans, "No Title," 2017. https://eightthirtyfour.com/data (accessed Jan. 09, 2020).

[11] Y. Reich and S. J. Fenves, "The Formation and Use of Abstract Concepts in Design," in *Concept Formation*, Elsevier, 1991, pp. 323–353.

[12] M. Kates, "Player Motion Analysis: Automatically Classifying NBA Plays," Massachusetts Institute of Technology, 2014.

[13] D. H. Fisher, M. J. Pazzani, and P. Langley, *Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufmann, 2014.

[14] I. H. Witten and E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations," *ACM Sigmod Rec.*, vol. 31, no. 1, pp. 76–77, 2002.

[15] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine Learning for Internet of Things Data Analysis: A Survey," *Digit. Commun. Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[16] A. Nistala, "Using Deep Learning to Understand Patterns of Player Movement in Basketball," Massachusetts Institute of Technology, 2018.

[17] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[18] D. Lutz, “A Cluster Analysis of NBA Players,” in *Proceedings of the MIT Sloan Sports Analytics Conference, Boston, MA, USA.*, 2012, vol. 24, p. 2016.

[19] A. Franks, A. Miller, L. Bornn, K. Goldsberry, and others, “Characterizing the Spatial Structure of Defensive Skill in Professional Basketball,” *Ann. Appl. Stat.*, vol. 9, no. 1, pp. 94–121, 2015.

[20] J. Sampaio, T. McGarry, J. Calleja-González, S. Jiménez Sáiz, X. i del Alcázar, and M. Balciunas, “Exploring Game Performance in the National Basketball Association Using Player Tracking Data,” *PLoS One*, vol. 10, no. 7, p. e0132894, 2015.

[21] J. Sampaio, E. J. Drinkwater, and N. M. Leite, “Effects of Season Period, Team Quality, and Playing Time on Basketball Players’ Game-related Statistics,” *Eur. J. Sport Sci.*, vol. 10, no. 2, pp. 141–149, 2010.

[22] M. Teramoto, C. L. Cross, R. H. Rieger, T. G. Maak, and S. E. Willick, “Predictive Validity of National Basketball Association Draft Combine on Future Performance,” *J. Strength \& Cond. Res.*, vol. 32, no. 2, pp. 396–408, 2018.

[23] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[24] C. Ulas and M. Cetin, “Incorporation of a language model into a brain computer interface based speller through HMMs,” 2013.

[25] X. Wu *et al.*, “Top 10 algorithms in data mining,” *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.

[26] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly Media, Incorporated, 2019.

[27] J. Huang, J. Lu, and C. X. Ling, “Comparing Naive Bayes, Decision Trees, and SVM with AUC and Accuracy,” 2003.

[28] R. D. S. Raizada and Y.-S. Lee, “Smoothness without Smoothing: Why Gaussian Naive Bayes Is Not Naive for Multi-Subject Searchlight Studies,” *PLoS One*, vol. 8, no. 7, p. e69566, Jul. 2013, doi: 10.1371/journal.pone.0069566.

[29] F. Bre, J. M. Gimenez, and V. D. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using artificial neural networks,” *Energy Build.*, vol. 158, pp. 1429–1441, Jan. 2018, doi: 10.1016/j.enbuild.2017.11.045.

[30] H. Leung and S. Haykin, “The Complex Backpropagation Algorithm,” *IEEE Trans. Signal Process.*, vol. 39, no. 9, pp. 2101–2104, 1991, doi: 10.1109/78.134446.

[31] F. Murtagh, “Multilayer perceptrons for classification and regression,” *Neurocomputing*, vol. 2, no. 5–6, pp. 183–197, Jul. 1991, doi: 10.1016/0925-2312(91)90023-5.

[32] F. Chollet and others, “Keras.” GitHub, 2015, [Online]. Available:

https://github.com/fchollet/keras.

[33] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 2016, pp. 265–283.

[34] J. Donahue *et al.*, “Decaf: A Deep Convolutional Activation Feature for Generic Visual Recognition,” in *International conference on machine learning*, 2014, pp. 647–655.

[35] Y. Li, W. Li, V. Mahadevan, and N. Vasconcelos, “VLAD3: Encoding Dynamics of Deep Features for Action Recognition,” Jun. 2016.

[36] C. E. Metz, “Basic principles of ROC analysis,” *Semin. Nucl. Med.*, vol. 8, no. 4, pp. 283–298, 1978, doi: 10.1016/S0001-2998(78)80014-2.

[37] B. Azhagusundari and A. S. Thanamani, “Feature Selection based on Information Gain,” *Int. J. Innov. Technol. Explor. Eng.*, no. 2, p. 18, 2013, doi: 10.1016/j.asoc.2008.05.006.

[38] L. B. Godfrey, “Parameterizing and Aggregating Activation Functions in Deep Neural Networks,” 2018. Accessed: Apr. 01, 2021. [Online]. Available: http://scholarworks.uark.edu/etdhttp://scholarworks.uark.edu/etd/2655.

[39] L. B. Dan Cervone and K. Goldsberry, “NBA Court Realty,” 2016.

VITA

DEMBÉ KOI STEPHANOS

Education:	M.S. Computer Science, East Tennessee State University, Johnson City, Tennessee, 2021
	B.S. Philosophy Minor in Computing, East Tennessee State University, Johnson City, Tennessee, 2015
	Flatiron School Full-Stack Web Development Bootcamp, New York, New York, 2018
Professional Experience:	Software Engineer, Scientific Financial Systems; Boston, Massachusetts, 2021- Present
	Graduate Associate, East Tennessee State University, College of Business and Technology, 2019-2021
	Software Dev Intern, Red Berry Innovations, Springfield, Nebraska, 2020
	Political Technology Specialist, DigiDems, Los Angeles, California, 2018
Publications:	D. K. Stephanos, G. Husari, B. Bennett and E. Stephanos, "Machine Learning Predictive Analytics for Player Movement Prediction in NBA: Applications, Opportunities, and Challenges," *ACM Southeast,* 2021.
Honors:	Phi Kappa Phi, 2021